UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA
INGENIERÍA DE TELECOMUNICACIONES

PROYECTO DE FIN DE CARRERA

Autenticación y administración centralizada en sistemas VoIP con Asterisk y LDAP

AUTORA: María Ana Ruiz Cañamero
TUTOR: Dr. Mario Muñoz Organero
Junio de 2010

ÍNDICE DE CONTENIDOS

AGRADECIMIENTOS ……5
RESUMEN……..6
1. INTRODUCCIÓN Y OBJETIVOS DEL PROYECTO....8
1.1. Introducción....8
1.2. Objetivos del Proyecto....9
1.3. Desarrollo del Proyecto....10
1.4. Organización del documento....11
2. TECNOLOGÍAS EMPLEADAS Y ESTADO DEL ARTE....12
2.1. LDAP....13
2.1.1. Modelos de LDAP....16
2.1.1.1. Modelo de información....16
2.1.1.2. Modelo de nombrado....19
2.1.1.3. Modelo funcional....20
2.1.1.4. Modelo de seguridad....21
2.2. Comparativa de implementaciones de LDAP....22
2.3. Asterisk....24
2.3.1. Historia y desarrollo de Asterisk....24
2.3.2. Características de Asterisk....26
2.3.3. Organización y versiones de Asterisk....27
2.3.4. Asterisk en LDAP....29
2.4. Protocolos de VoIP....30
2.4.1. Protocolo SIP....30
2.4.1.1. La identificación de usuarios en el direccionamiento SIP....33
2.4.1.2. Ejemplo de protocolo SIP....34
2.4.1.3. SIP y NAT....35
2.4.2. Comparativa de otros protocolos de VoIP con SIP....37
2.4.2.1. H.323....37
2.4.2.2. IAX....43
2.5. Codecs de compresión de datos de voz....44
2.5.1. G711: aLaw y uLaw....44
2.5.2. g.729....44
2.5.3. GSM....45
2.6. Tecnologías para implementar una aplicación Web: ECLIPSE y MAVEN....46
2.7. Librerías de cliente LDAP para Java: JLDAP....47
2.7.1. JLDAP....48
2.8. Análisis de productos similares en el mercado....51
3. DESCRIPCIÓN E IMPLEMENTACIÓN DEL SISTEMA....52
3.1. Requisitos de la aplicación a desarrollar....53
3.2. Características de la aplicación base....55
3.3. Componentes y descripción de la aplicación....56
3.4. Estructura de directorios y archivos....57
3.5. Tablas de la Base de Datos empleadas....59
3.6. Nuevos métodos desarrollados....60
3.7. Problemas encontrados y soluciones propuestas....70

4. PRUEBAS 73
4.1. Fase 1: Inicialización del directorio LDAP en la centralita telefónica 74
4.2. Fase 2: Actividades cotidianas de la centralita telefónica 78
4.2.1. Administración de los usuarios de la centralita 78
4.2.1.1. Creación de usuarios en la centralita telefónica 79
4.2.1.1.1. Creación de grupos de usuarios 79
4.2.1.1.2. Creación de usuarios. Dos casos: si el usuario aún no existe en LDAP o si el usuario ya existe en LDAP 86
4.2.1.2. Modificación de usuarios 93
4.2.1.3. Borrado de usuarios 93
4.2.2. Autenticación en el inicio de llamadas 94
5. MEMORIA ECONÓMICA Y PRESUPUESTO. 97
6. CONCLUSIONES Y POSIBLES MEJORAS. 99
6.1. Conclusiones 99
6.2. Mejoras futuras 100
7. BIBLIOGRAFÍA Y REFERENCIAS 101
8. ACRÓNIMOS 103
ANEXOS 105
1. Código de los nuevos métodos desarrollados 105
2. Manual de instalación de las herramientas del desarrollador 136
2.1. Manual de instalación de Eclipse y Maven 136
2.2. Descargar e instalar el JBOSS en el PC del desarrollador 141
2.3. Configurar Maven2 146
3. Manual de instalación de VMware Player, Debian y OpenLDAP 149
4. Esquemas de LDAP 185
4.1. asterisk.schema 186
4.2. java.schema 195
5. Servidor conteniendo la instalación de la aplicación de gestión de usuarios 197

Agradecimientos

Quiero agradecer a mi familia todo el apoyo y el ánimo que me ha dado tanto durante el tiempo de la carrera como durante el tiempo dedicado al proyecto. Jose Manuel, Ana María, Raquel, Elena, Victor y Rafa, este proyecto es también vuestro.

Muchas gracias también a toda la gente de Intecdom que me ha ayudado a resolver mis dudas y me ha orientado. Diego, Andrés y Roberto, gracias por el tiempo que me habéis dedicado.

Gracias a mis compañeros de carrera, con los que tanto tiempo he pasado en clases, laboratorios y prácticas, que han sido la base del proyecto personal que hoy cierra una fase. Ruth, María, Raquel, Natalia, Dani, Miguel, Javi, Luis, Carlos, Jose, Charo, Santi, Pablo, y tantos más que hemos recorrido juntos este camino.

Y gracias también a Mario, mi tutor en la UC3M, por su apoyo, su consejo, y por su actitud siempre positiva y de ánimo.

Muchas gracias a todos

Resumen

Este proyecto nace como una ampliación de funcionalidad de un desarrollo comercializable de empresa consistente en una centralita telefónica basada en software Asterisk. Dicha centralita telefónica basa el almacenamiento y autenticación de sus usuarios en una base de datos PostgreSQL, en la cual se ha creado una estructura de tablas de datos que se utilizan conjuntamente con la estructura de ficheros de configuración de Asterisk para la realización de llamadas telefónicas de voz sobre IP.

La empresa Intecdom S.L. ha desarrollado la centralita telefónica denominada IRI basada en el software libre Asterisk, en PostgreSQL, y en otros elementos que se detallarán a lo largo del proyecto. Esta empresa me propuso realizar un desarrollo para integrar su centralita IRI con directorios LDAP, para ofrecer de esta manera un producto más versátil que pudiera ser utilizado en distintos escenarios de negocio en los existiese un directorio LDAP o en los que se quisiera instalar uno.

El reto de integrar la centralita IRI con LDAP para el almacenado y autenticación de usuarios ha supuesto un análisis profundo de los siguientes puntos:

- Estudio de la tecnología LDAP, desde los aspectos más básicos hasta los más avanzados:
    - Qué es un directorio y en qué se diferencia de una base de datos
    - Qué operaciones básicas se pueden realizar en él y cómo se pueden utilizar estas operaciones para verificar, almacenar, y autenticar los usuarios de telefonía
    - Cuáles son las reglas que definen la estructura del directorio y cómo integrar nuevos elementos que estructuren al grupo de usuarios de telefonía
    - Qué elementos técnicos permiten la comunicación en tiempo real de las solicitudes de verificación, modificación, creación y autenticación de la centralita telefónica hacia el directorio
    - Aspectos de seguridad y calidad
- Estudio de la centralita telefónica existente:
    - Cómo se gestionan los usuarios en la centralita y qué operaciones pueden realizarse sobre ellos
    - Cómo se estructura la base de datos en la que se almacenan los usuarios y su atributos, y cómo se almacenan otros atributos de grupos de usuarios, perfiles y características de la centralita
    - El funcionamiento de los ficheros de configuración de Asterix y los módulos que permiten integrar nuevas funcionalidades

- Estudio del entorno de programación que permite ampliar los desarrollos actuales de la centralita
    - El programa Eclipse en el que se ha desarrollado el código Java en el que se basa la implementación de la centralita y de su interfaz de usuario
    - Funcionalidad del programa Maven 2 para gestionar el ciclo de vida de la aplicación y añadir nuevos desarrollos
    - JLDAP y las correspondientes librerías de Novell para integrar el desarrollo de la centralita en Java con el directorio LDAP

Todos estos elementos han permitido desarrollar la ampliación del producto solicitado, del cual se ha realizado una batería de pruebas para comprobar su correcto funcionamiento.

1. INTRODUCCIÓN Y OBJETIVOS DEL PROYECTO

1.1. Introducción

La idea de este proyecto fin de carrera surge ante la necesidad por parte de una empresa tecnológica de integrar los servicios de telefonía IP basados en Asterisk y la autenticación contra LDAP, concretándose en la modificación de una aplicación perteneciente a dicha empresa.

El producto software desarrollado se integra en una aplicación Web que facilita la gestión de usuarios del sistema de telefonía, pudiendo acceder a la aplicación tanto un usuario administrador que realiza tareas de creación, configuración y mantenimiento de usuarios, como los usuarios concretos para acceder y actualizar su información personal. Las modificaciones introducidas en el producto habilitan la posibilidad de elegir en el momento de instalación del mismo si la autenticación se quiere realizar contra una base de datos local, tal como se hacía en el producto original, o si se realiza contra un servidor LDAP.

La necesidad de incluir la autenticación contra LDAP viene determinada por el hecho de que el uso de este tipo de directorios está cada vez más extendido y por lo tanto la integración de ambos productos ofrecería una herramienta final de gran utilidad.

Este proyecto se ha desarrollado como proyecto en empresa durante mi estancia de prácticas en InTecDom, una empresa que desarrolla y distribuye entre otros productos soluciones de telefonía IP basadas en Asterisk y en el protocolo de comunicaciones SIP. El trabajo consiste en el desarrollo e integración de una aplicación ya existente en la empresa con LDAP. Dicha aplicación, llamada IRI, está programada en lenguaje Java utilizando el entorno de programación Eclipse sobre un sistema operativo Linux, e integra el software libre de telefonía IP de Asterisk con una base de datos PostgreSQL. El reto es por tanto desarrollar dentro del mismo entorno el software necesario para integrar LDAP y pasar por lo tanto de la autenticación basada en base de datos a una autenticación basada en directorio LDAP.

Tras revisar el estado del arte, se pudo comprobar que han aparecido diversos desarrollos que facilitan la integración de la programación en Java con LDAP, así como la integración en LDAP de esquemas estandarizados para la utilización de Java y de Asterisk. Una vez definido el diseño del producto, y utilizando el entorno de desarrollo de la interfaz Web existente, hay que programar tanto las nuevas páginas como las modificaciones en las páginas existentes, necesarias para completar la integración del sistema de telefonía IP con LDAP.

1.2. Objetivos del Proyecto

Los objetivos este documento son los siguientes: presentar las tecnologías que forman parte del producto a desarrollar, explicar el proceso llevado a cabo para su realización, y explicar la funcionalidad final del mismo. Para ello, el documento no solo contendrá una descripción de la aplicación desarrollada, con los fragmentos de código más relevante incluido en los anexos de este proyecto, sino que además planteará casos de uso utilizados en el paratado de pruebas que pongan de manifiesto su funcionalidad y utilidad.

Es necesario destacar que tanto el código total como el producto generado pertenecen a la empresa propietaria del producto base, siendo público sólo el documento que aquí se presenta. Por lo tanto no se incluirá el código fuente del producto total, sino algunos fragmentos correspondientes a la integración con LDAP que es el objetivo de este proyecto.

Como se ha comentado, a lo largo del proyecto se realizará una descripción de las tecnologías utilizadas así como de la integración de las mismas para producir una solución final funcional. Esta descripción tiene un carácter didáctico y pretende familiarizar al lector con cada tecnología mencionada y con el estado del arte, lo cual permitirá enmarcar el producto su ámbito correspondiente, a partir de la descripción de las distintas tecnologías utilizadas en su realización.

De la misma manera, se detallarán los problemas prácticos encontrados durante la realización el proyecto, así como las soluciones adoptadas.

Finalmente se detalla la funcionalidad, requisitos, configuración e instalación del software desarrollado, tanto en el apartado de implementación y desarrollo como en los Anexos.

1.3. Desarrollo del Proyecto

En primer lugar ha sido necesario un amplio estudio de las tecnologías involucradas, en especial de LDAP, Asterisk y el entorno de programación Eclipse para Java. Para ello he realizado un estudio de la información recopilada mediante diversos medios, tanto bibliografía sobre los mismos como abundante información existente en Internet, en las páginas Web de los proyectos LDAP, Asterisk, Sun y Eclipse.

Respecto al lenguaje de programación utilizado, al realizarse una integración en una herramienta ya existente, la libertad para la elección de lenguajes o entornos alternativos está limitada y condicionada a la línea de desarrollo existente, adaptando los cambios en cuanto al entorno de programación y sus funcionalidades. Por ejemplo, la versión de la herramienta Eclipse tuvo que ser actualizada, y se integró a un repositorio común denominado Maven2 utilizado para centralizar las modificaciones y adiciones de diversos desarrolladores en paralelo sobre el mismo producto.

El desarrollo del proyecto se puede desglosar en las siguientes fases:

1. Estudio del estado del arte de las tecnologías implicadas
2. Definición de los requisitos de la nueva herramienta
3. Estudio de la herramienta existente
4. Desarrollo de la herramienta
5. Preparación de la maqueta instalable del producto final
6. Batería de Pruebas
7. Análisis final de trabajo y costes
8. Realización de la documentación y revisión de todas las fases anteriores

1.4. Organización del documento

El presente documento se organiza principalmente siguiendo la definición de las fases del proyecto descritas anteriormente.

El capítulo 2 contiene una extensa documentación sobre las tecnologías empleadas, su estado del arte y una comparativa con otras tecnologías de su ámbito.

El capítulo 3 contiene una descripción del sistema, tanto de la aplicación de partida, como de la integración con LDAP que se va a realizar, qué partes del sistema existente se ven afectadas y cómo se modifica la funcionalidad del producto final. Este capítulo también incluye la implementación realizada, explicando los métodos desarrollados. Este apartado sirve de guía para poder revisar posteriormente la codificación de las funciones que comunican los datos introducidos por el usuario desde la aplicación Web con LDAP y Asterisk, la cual está incluida en los anexos.

En el capítulo 4 se describen las pruebas realizadas sobre el producto final para confirmar su correcta funcionalidad, repasando por tanto los casos de uso y escenarios de utilización del producto.

El capítulo 5 trata sobre la memoria económica, en él se revisan tanto los materiales utilizados como el esfuerzo invertido en la realización del proyecto. Con todos estos datos se realiza una valoración económica de los costes incurridos.

El capítulo 6 contiene un repaso final de las conclusiones y posibles mejoras. Es una reflexión sobre el trabajo realizado, así como la discusión de posibles líneas de desarrollo futuro relacionadas con el contenido de este proyecto.

Finalmente los capítulos 7 y 8 recogen por un lado las Referencias y Bibliografía utilizada, y por otro los Acrónimos, tan necesarios en el mundo tecnológico actual.

Por último se incluyen los anexos que completan el trabajo realizado, añadiendo el código implementado, dos manuales de instalación y un último anexo con los esquemas de LDAP añadidos a los esquemas básicos para poder realizar el proyecto. El primer manual de instalación es para el desarrollador y pretende resumir los pasos de instalación necesarios para poner a punto las herramientas requeridas para la realización del proyecto. El segundo manual es una guía de instalación de todos los elementos necesarios para instalar y poner a punto un directorio LDAP.

2.TECNOLOGÍAS EMPLEADAS Y ESTADO DEL ARTE

Hoy en día existen múltiples soluciones de telefonía IP basadas en Asterisk. La licencia de código abierto del *software* facilita que multitud de empresas de diverso tamaño y origen se dediquen al desarrollo de soluciones adaptadas a las necesidades del mercado.

Las diversas posibilidades para implementar la autenticación de usuarios de Asterisk presentan distintos niveles de complejidad. El más simple sería la utilización directa de los ficheros básicos de configuración de Asterisk; la solución alternativasería la utilización de una base de datos local en la que están guardados los datos de usuarios sy a la que se consulta cada vez que se requiere autenticación; y por último la autenticación directa de usuarios creados en un directorio situado en un servidor LDAP externo.

Podemos encontrar un gran número de soluciones particulares que implementan centralitas telefónicas basadas en todas las opciones descritas anteriormente, profundizando este proyecto en el análisis e implementación de las modificaciones necesarias para pasar de una solución de segundo tipo a una de tercer tipo, sin dejar de lado todo el análisis necesario sobre la aplicación de partida y las tecnologías involucradas en su realización.

A lo largo de este apartado se realizará una breve introducción a modo de repaso de todos los conceptos necesarios para comprender el proyecto desarrollado y el ámbito en el que se enmarca.

Se describirán los conceptos básicos y criterios de elección sobre las siguientes tecnologías:

- LDAP
- Asterisk
- Protocolos de VoIP: SIP
- Codecs
- Arquitecturas de aplicación basadas en lenguajes de programación: Eclipse
- Librerías de Cliente LDAP para Java: JLDAP

Una vez definidos estos conceptos y entendida la situación actual de las soluciones basadas en las tecnologías anteriores, se procederá al desarrollo del producto final basándose en las mismas.

2.1. LDAP

LDAP son las siglas en inglés correspondientes a: Lightweight Directory Access Protocol.

LDAP es un estándar abierto para el acceso a servicios de directorios basados en X.500, que se ejecuta sobre TCP/IP o sobre otros servicios de transferencia de datos orientados a conexión. LDAP está especificado en el siguiente protocolo del IETF: "Lightweight Directory Access Protocol (LDAP) Technical Specification Road Map" RFC4510.

El uso de los directorios en el mundo empresarial se ha extendido. Esta expansión viene justificada entre otros factores por el hecho de que hoy en día se tiende a ofrecer a los clientes unos servicios cada vez más personalizados. Es necesario para ello el uso de bases de datos que almacenen esta información personalizada y la utilización de herramientas que faciliten su acceso y puedan ser fácilmente mantenidas.

Un directorio es una base de datos especializada. Un directorio puede definirse más detalladamente como una colección jerárquica de objetos y de los atributos de esos objetos. Un objeto puede contener diversos atributos, y los atributos pueden tener uno o más valores.

Un directorio incluye los siguientes elementos básicos que permiten que la información contenida en él sea accesible a los usuarios:

- La información contenida en el directorio
- Los servidores software que contienen esta información
- Los clientes software por medio de los cuales los usuarios acceden a la información
- El hardware sobre el que corren estos clientes y servidores
- El software de soporte, como por ejemplo el sistema operativo y los controladores de los dispositivos.
- La infraestructura de red que conecta a los clientes con los servidores, y a los servidores entre ellos
- Las normas mediante las que se decide quién puede acceder y actualizar el directorio, qué se puede guardar en él, etc.
- Los procedimientos con los que se mantiene y monitoriza el directorio
- El software utilizado para mantener y monitorizar el directorio

LDAP es un directorio de propósito general, basado en estándares. Estos directorios sirven para una gran variedad de de aplicaciones.

Una de las primeras preguntas que suele surgir al hablar de directorios es, ¿cuál es la diferencia entre una base de datos y un directorio? Un directorio es un tipo específico de base de datos. Las características que definen a un directorio son:

- La tasa de acceso de lectura es típicamente mayor que la de escritura en un directorio que en una base de datos
- La facilidad para extender los atributos de la información que contiene es mayor
- Están más distribuidos (mayor escala de distribución)
- Son replicados más a menudo y a mayor escala, para proporcionar una mayor fiabilidad del sistema, más rápida disponibilidad de los recursos, y mayor eficiencia en su uso. El sincronismo total entre los datos de las réplicas no es tan importante como en las bases de datos.
- Tienen diferentes características de funcionamiento. La eficiencia, medida en número de transacciones que pueden realizarse por segundo, es mayor en los directorios
- Es más importante que soporten estándares, ya que cualquier aplicación propietaria debería ser capaz de interoperar con el directorio

La información es ordenada y organizada en los directorios para que pueda ser fácilmente encontrada por los usuarios.

Los directorios del mundo de las redes de ordenadores son directorios online. Se diferencian de los directorios *offline* tradicionales (páginas amarillas, guía TV, catálogo biblioteca) en que los directorios online poseen las siguientes características:

- Dinámicos: tienen mayor capacidad de actualización, los datos exactos se actualizarán al momento.
- Flexibles:
    - En contenido: se pueden extender (añadir categorías extra de datos) sin rediseñarse. La nueva información es accedida bajo demanda.
    - En organización: se pueden hacer búsquedas bajo distintos criterios, con garantía de consistencia de los datos ya que no hay información duplicada. También se pueden hacer búsquedas de términos análogos, y hacer distintas búsquedas simultáneamente.
- Seguros: se puede controlar el acceso a la información ya que se autentica a los usuarios. Hay listas de control de acceso (ACLs).
- Personalizados:
    - En cuanto al servicio de entrega de información a los usuarios, cuando acceden al directorio, que se puede adaptar a sus preferencias particulares.

- En cuanto al tratamiento de la información contenida en el directorio: se puede guardar un perfil de usuario, para recordar sus preferencias

Los directorios contienen unos elementos denominados esquemas. Estos esquemas definen el tipo de información que se almacena en el directorio, las reglas que debe cumplir dicha información y cómo se realizan las operaciones de búsqueda sobre los datos. Los esquemas pueden ser modificados para cubrir las necesidades específicas de un directorio.

Los esquemas básicos contienen un extenso conjunto de clases de objeto y atributos con los que se cumplen todas las necesidades de la mayoría de organizaciones. Se modelan siguiendo el estándar X.500 de la *International Standards Organisation* (ISO) para los servicios de directorio.

Los esquemas son modificables y ampliables, no obstante en este proyecto no ha sido necesario modificar los esquemas existentes, dada la extensión en la integración de los directorios LDAP con múltiples aplicaciones y lenguajes de programación. En concreto ha sido necesario añadir los esquemas oficiales de LDAP correspondientes a la integración con Asterisk y con java, se añaden ambos esquemas en el anexo al final del documento.

2.1.1. Modelos de LDAP

LDAP puede ser definido en base a estos cuatro modelos:

- Modelo de información: describe la estructura de la información guardada en el directorio
- Modelo de nombrado: describe cómo se identifica la información, y cómo se organiza en base a su nombre
- Modelo funcional: describe las operaciones que se pueden realizar sobre la información
- Modelo de seguridad: describe cómo está protegida la información respecto a accesos no autorizados

A continuación profundizaremos en estos cuatro modelos.

2.1.1.1. Modelo de información

La unidad básica de información almacenada en el directorio es la entrada, en inglés *entry*, que representa a un objeto.

Cada entrada está compuesta de un conjunto de atributos. Cada atributo es de un tipo determinado, y tiene asignado uno o varios valores. El tipo define la clase de información que puede almacenar el atributo, y los valores son la información propiamente dicha.

En la siguiente figura podemos ver una representación gráfica de los elementos definidos.

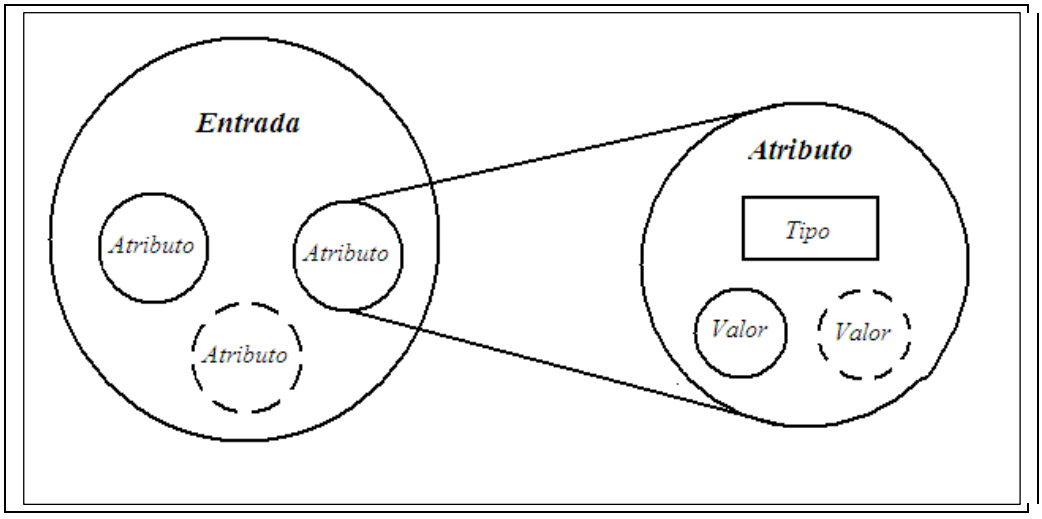


Figura 1. Elementos del modelo de información de un directorio

Los atributos tienen asociada una determinada sintaxis y un comportamiento, de este modo quedarán definidos los tipos de datos que se pueden almacenar como valores, y el comportamiento de los mismos en las operaciones realizadas sobre el directorio.

Algunos ejemplos de sintaxis son:

- bin: información binaria
- ces: tener en cuenta las mayúsculas en operaciones de comparación
- cis: no tener en cuenta las mayúsculas en operaciones de comparación

A continuación se muestra un pequeño ejemplo de una entrada de directorio. Como se puede apreciar, cada atributo está relacionado con un alias, para referenciarlo de forma breve.

Atributo, Alias	**Sintaxis**	**Descripción**	**Ejemplo**
commonName, cn	cis	Nombre común	Ana Ruiz
Surname, sn	cis	Apellido	Ruiz
Organization, o	cis	organización	Intecdom
organizationalUnitName, ou	cis	Nombre del departamento	Desarrollo
Mail	ces	Dirección de correo electrónico	ana.ruiz@intecdom.es

Los esquemas definen los atributos obligatorios y opcionales de los objetos que se van a almacenar en el directorio. El siguiente fragmento del esquema *core.schema* muestra la definición de atributos:

```
# system schema
#attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' )
#       DESC 'RFC2256: common name(s) for which the entity is known by'
#       SUP name )

attributetype ( 2.5.4.4 NAME ( 'sn' 'surname' )
        DESC 'RFC2256: last (family) name(s) for which the entity is known by'
        SUP name )
attributetype ( 2.5.4.10 NAME ( 'o' 'organizationName' )
        DESC 'RFC2256: organization this object belongs to'
        SUP name )
```

```
attributetype ( 2.5.4.11 NAME ( 'ou' 'organizationalUnitName' )
        DESC 'RFC2256: organizational unit this object belongs to'
        SUP name )
attributetype ( 0.9.2342.19200300.100.1.3
        NAME ( 'mail' 'rfc822Mailbox' )
        DESC 'RFC1274: RFC822 Mailbox'
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256}

# system schema
#attributetype ( 2.5.4.35 NAME 'userPassword'
#       DESC 'RFC2256/2307: password of user'
#       EQUALITY octetStringMatch
#       SYNTAX 1.3.6.1.4.1.1466.115.121.1.40{128} )
```

La RFC 2252 define todas las normas de sintaxis que deben cumplir los esquemas LDAP para permitir la interoperabilidad entre distintos esquemas:

- cómo debe ser la definición de un objeto y de un atributo (etiquetas NAME, DESC)
- qué tipos de datos se pueden definir (etiqueta SYNTAX)
- qué reglas se pueden emplear (etiquetas SUBSTR, ORDERING, EQUALITY)

Entre los objetos existe el concepto de herencia, lo cual determina la estructuración de los datos. Unos objetos pueden derivar de otros heredando sus atributos y añadiendo nuevos atributos a los ya definidos en el padre. Por ejemplo, la clase *organizationalPerson* hereda de la clase *Person*: un objeto que implemente la clase organizationalPerson puede contener tanto los atributos de Person (cn, sn, etc.) como los atributos de organizationalPerson (número de empleado, etc.)

El atributo *ObjectClass* debe estar contenido en todas las entradas, ya que define la clase de objeto que representa la entrada, así como los posibles atributos que contendrá. Cuando una entrada pertenece a una clase que no puede heredar de otras, entonces su valor es *top*. Siempre debe haber al menos una entrada de este tipo, ya que las entradas deben contener tanto la clase que implementan como las clases superiores a ésta de las cuales heredan.

2.1.1.2. Modelo de nombrado

Este modelo define cómo se organizan y se identifican los datos. Una vez creada la estructura de árbol del directorio, el modelo de nombrado nos indica cómo referenciar las entradas.

Cada entrada tiene un identificador único llamado DN, del inglés *Distinguished Name*. El DN consiste en una secuencia de partes más pequeñas llamadas RDN, del inglés *Relative Distinguished Name*. Típicamente las entradas se organizan en una estructura de árbol denominada DIT, *Directory Information Tree*. En una estructura de árbol no hay ninguna entrada suelta o independiente, y tan sólo la entrada raíz no tiene padre.

A continuación se muestra un ejemplo de árbol de directorio

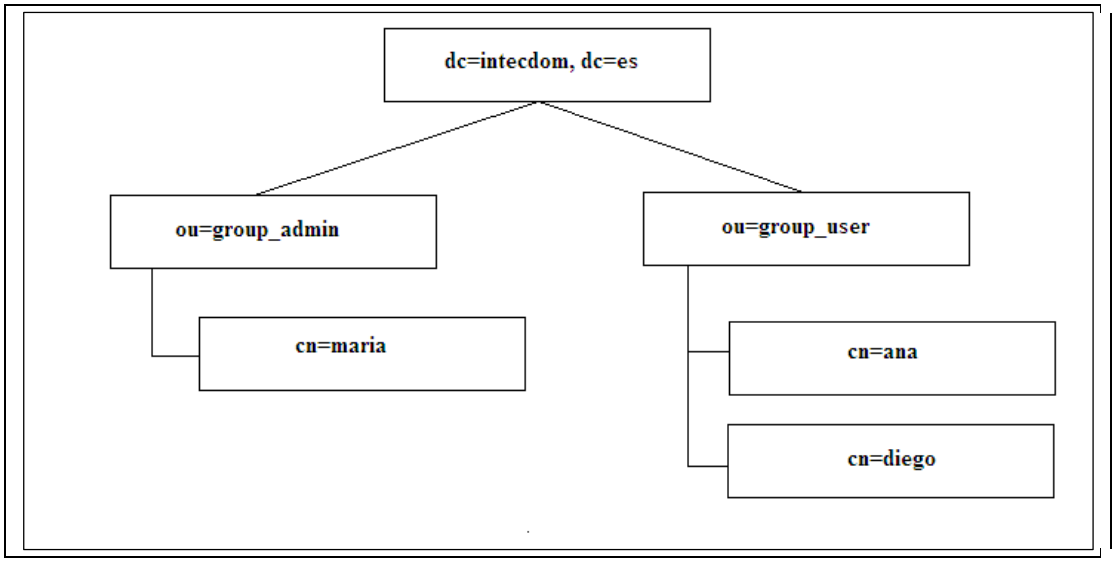


Figura 2 Ejemplo de árbol de directorio

En el ejemplo anterior, tenemos un modelo de organización con dos departamentos, los cuales contienen a su vez objetos de tipo persona (*person*). El DN de una persona en esta organización sería por ejemplo el siguiente:

DN: cn=ana, ou=group_user, dc=intecdom, dc=es

El tamaño de los árboles de directorio es variable, y con frecuencia es demasiado grande para ser almacenado en un solo servidor, por lo cual la información es distribuida entre distintos servidores. Los distintos servidores se encuentran enlazados mediante referencias (*referals*) que se definen como clases. De esta manera si un cliente realiza una petición sobre una entrada que no está almacenada en el servidor, la respuesta será una referencia a otro servidor pudiendo el cliente continuar la búsqueda en el nuevo servidor. Este método de referencias proporciona una gran flexibilidad y permite el balanceo de carga entre distintos servidores.

2.1.1.3. Modelo funcional

Este modelo es el encargado de definir las operaciones que se pueden realizar sobre los datos del directorio. A grandes rasgos, hay tres tipos básicos de operaciones: de consulta, de actualización y de autenticación:

- Operaciones de consulta: permiten realizar búsquedas de información almacenada en el directorio.
    - *Search*: realiza una búsqueda de entradas que cumplan las especificaciones indicadas (punto de inicio de búsqueda, profundidad, filtros, formato de respuestas, etc.)
    - *Compare*: determina si una entrada tiene un valor concreto en un atributo específico.
- Operaciones de actuación: permiten añadir, borrar, renombrar y modificar entradas del directorio:
    - *Add*: añade nuevas entradas al directorio, que se especifican mediante el DN de la entrada y sus atributos con sus respectivos valores. Para que una nueva entrada sea añadida correctamente debe cumplirse que el padre de la entrada exista en el directorio, que no haya otra entrada con el mismo DN, y que se cumplan los requisitos especificados en el esquema.
    - *Delete*: elimina una entrada existente en el directorio dado su DN. La entrada a borrar no debe tener hijos.
    - *Rename*: modifica el DN de una entrada existente en el directorio.
    - *Modify*: modifica los atributos de una entrada existente.
- Operaciones de autenticación y control: permiten autenticar al los clientes LDAP y controlar las sesiones abiertas contra el servidor
    - *Bind*: autentica al cliente frente al directorio. Hay varios tipos de autenticación:
        - Sesión anónima: sólo permite realizar búsquedas (ldapsearch)
        - Sesión autenticada: empleando una contraseña
        - Sesión cifrada: utilizando los mecanismos de SASL
    - *Unbind*: cierra la sesión contra LDAP
    - *Abandon*: indica al servidor que el cliente abandona la operación actual.

2.1.1.4. Modelo de seguridad

Se basa en la operación *Bind* definida en el apartado anterior. Primero definiremos unos términos relacionados con la seguridad que servirán para comprender los conceptos:

- Autenticación: es el proceso de verificar la identidad digital del remitente de una comunicación como una petición para conectarse. El remitente puede ser una persona que usa un ordenador, un ordenador por sí mismo o un programa del ordenador. En una Web de confianza, "autenticación" es un modo de asegurar que los usuarios son quién ellos dicen que ellos son.
- Integridad: la verificación de que un mensaje ha sido o no manipulado (sea por incidente o por accidente).
- Confidencialidad: La confidencialidad ha sido definido por la ISO en la norma ISO-17799 como "garantizar que la información es accesible sólo para aquellos autorizados a tener acceso"
- Autorización: asegurar que los recursos del sistema sólo sean utilizados por aquellos usuarios a los que se les ha concedido autorización para ello, es decir, asegurar que los recursos sólo sean utilizados por quienes tienen permiso.

Los mecanismos de seguridad son distintos según la versión de LDAP. La última versión de LDAP disponible, y que ha sido la utilizada en este proyecto es la versión 3, por lo que explicaremos brevemente los mecanismos de seguridad correspondientes:

- Sesión anónima: No utiliza autenticación. Sólo permite realizar búsquedas, ya que se asume que ésta es la única operación que no tiene ninguna restricción de acceso a los datos. Se utiliza dejando vacíos los campos DN y *password* en la llamada al *bind*, con lo cual el servidor LDAP entiende que el usuario es anónimo y establece los controles de acceso para ese rol.
- Sesión autenticada: empleando un DN y una contraseña se consigue una autenticación básica. Estos datos se envían en texto claro, sin cifrar. El servidor autentica al cliente si existe el DN y la contraseña coincide con la almacenada.
- Sesión cifrada: utilizando los mecanismos de SASL, que es un marco para autenticación y autorización en protocolos de Internet. Separa los mecanismos de autenticación de los protocolos de la aplicación permitiendo a cualquier protocolo de aplicación que use SASL usar cualquier mecanismo de autenticación soportado por SASL. A pesar de que mediante SASL sólo se maneja la autenticación y se requieren otros mecanismos, como por ejemplo TLS, para cifrar el contenido que se transfiere, SASL proporciona medios para un uso negociado del mecanismo elegido. Primero se negocia en claro acerca del mecanismo a utilizar, y una vez acordado éste la conexión es segura. Para autenticar se continúa utilizando el DN y la contraseña, pero ahora se pueden enviar cifrados. TLS utiliza los mecanismos de clave pública/privada para el cifrado. SASL no estaba presente en la versión 2 de LDAP y se añadió en la versión 3.

2.2. Comparativa de implementaciones de LDAP

Existen diversas implementaciones y aplicaciones reales del protocolo LDAP. A continuación se detallan las principales (fuente de las descripciones: Wikipedia)

- Active Directory: Active Directory es el nombre utilizado por Microsoft (desde Windows 2000) como almacén centralizado de información de uno de sus dominios de administración. Bajo este nombre se encuentra realmente un esquema (definición de los campos que pueden ser consultados) LDAP versión 3, lo cual permite integrar otros sistemas que soporten el protocolo. En este LDAP se almacena información de usuarios, recursos de la red, políticas de seguridad, configuración, asignación de permisos, etc.

  Esta implementación viene distribuida por defecto con Windows Server 2008 R2

- Novell Directory Services, también conocido como *eDirectory* es la implementación de Novell utilizada para manejar el acceso a recursos en diferentes servidores y computadoras de una red. Básicamente está compuesto por una base de datos jerárquica y orientada a objetos, que representa cada servidor, computadora, impresora, servicio, personas, etc. entre los cuales se crean permisos para el control de acceso, por medio de herencia. Esta implementación corre en diversas plataformas, por lo que puede adaptarse fácilmente a entornos que utilicen más de un sistema operativo.

- iPlanet - Sun ONE Directory Server: Basado en la antigua implementación de Netscape, iPlanet se desarrolló cuando AOL adquirió Netscape Communications Corporation y luego conjuntamente con Sun Microsystems comercializaron software para servidores, entre ellos el iPlanet Directory Server, su implementación de LDAP. Actualmente se denomina Sun ONE Directory Server.

- Red Hat Directory Server: Directory Server es un servidor basado en LDAP que centraliza configuración de aplicaciones, perfiles de usuarios, información de grupos, políticas, así como información de control de acceso dentro de un sistema operativo independiente de la plataforma.

  Forma un repositorio central para la infraestructura de manejo de identidad, Red Hat Directory Server simplifica el manejo de usuarios, eliminando la redundancia de datos y automatizando su mantenimiento.

- Apache Directory Server: Apache Directory Server (ApacheDS), es un servidor de directorio escrito completamente en Java por Alex Karasulu y disponible bajo la licencia de Apache Software, es compatible con LDAPv3 certificado por el Open Group, soporta otros protocolos de red tal como Kerberos y NTP, además provee procedimientos almacenados, triggers y vistas; características que están presente en las bases de datos relacionales pero que no estaban presentes en el mundo LDAP.

- Open DS: Basado en los estándares LDAPv3 y DSMLv2, OpenDS surgió como un proyecto interno de SUN, aunque posteriormente se puso a disposición de la comunidad. Está desarrollado en Java y precisa de un entorno de ejecución JRE (*Java Runtime Environment*) para funcionar. Es multiplataforma.

  La primera versión estable de Open DS fue liberada en julio de 2008.

- OpenLDAP: Se trata de una implementación libre del protocolo que soporta múltiples esquemas por lo que puede utilizarse para conectarse a cualquier otro LDAP.

  Tiene su propia licencia, la *OpenLDAP Public License*. Al ser un protocolo independiente de la plataforma, varias distribuciones GNU/Linux y BSD lo incluyen, al igual que AIX, HP-UX, Mac OS X, Solaris, Windows (2000/XP) y z/OS.

  OpenLDAP tiene cuatro componentes principales:

  - slapd - demonio LDAP autónomo.
  - slurpd - demonio de replicación de actualizaciones LDAP autónomo. Este componente ha sido sustituido por otro modo de replicación denominado syncrepl.
  - Rutinas de biblioteca de soporte del protocolo LDAP.
  - Utilidades, herramientas y clientes.

La última de las soluciones propuestas ha sido la elegida para ser implementada. Las características que han llevado a decidirse por OpenLDAP son las siguientes:

- Es una implementación libre, no hay que pagar la licencia por instalarlo. Esto disminuye los costes de realización del proyecto.
- Es un protocolo independiente de plataforma, lo cual le da gran flexibilidad de uso.
- Está muy extendido su uso, y posee una comunidad y foro de usuarios que da soporte a su instalación y mantenimiento. Esto facilita la tarea de ir resolviendo los errores que se presentan en la instalación, en la integración y durante el desarrollo del código.

2.3. Asterisk

Figura 3. Logo de Asterisk. Fuente: Asterix.org

Asterisk es un programa desarrollado bajo licencia de software libre que implementa una centralita telefónica con diversas funcionalidades.

Fue creado en 1999 por Mark Spencer, que continúa siendo hoy su principal desarrollador. El hecho de publicar el código fuente del programa bajo la Licencia Pública General de GNU (más conocida por su nombre en inglés: *GNU General Public License*) hace el código accesible al público en general permitiendo que otros programadores contribuyan a corregir errores, probar el código y añadir nuevas funcionalidades.

Asterisk fue desarrollado inicialmente para el sistema operativo Linux, sin embargo hoy en día existen versiones disponibles del software para la mayoría de sistemas operativos: BSD, MacOSX, Solaris, Microsoft Windows… No la plataforma nativa GNU/Linux es la que cuenta con mejor soporte de todas.

La versión de Asterisk utilizada para la implementación de este proyecto es la versión 1.6.1

2.3.1. Historia y desarrollo de Asterisk

El proyecto Asterisk comenzó en 1999 cuando su creador Mark Spencer, por aquel entonces estudiante de Ingeniería informática de la Universidad de Auburn, Alabama, creó la una empresa para dar soporte a usuarios de Linux: *Linux Support Services*. Debido al alto coste que suponía adquirir una Centralita telefónica para su negocio, Spencer comenzó a desarrollar código C para construir su propia centralita telefónica en un PC bajo Linux. Este hecho constituiría origen de lo que en el futuro sería Asterisk.

La empresa Linux Support Services pasaría a convertirse en Digium en 2002, centrando sus objetivos en el desarrollo oficial de Asterisk, contando con la colaboración de una amplia comunidad de desarrolladores surgida a raíz de la distribución del código fuente bajo la licencia de Software Libre.

Para hacer posible la integración del código generado por la comunidad de desarrolladores, además de las herramientas habituales en este tipo de proyectos tales como las listas de correo, chat y documentación en red, Asterisk utiliza dos elementos importantes:

- Un modelo de desarrollo basado el un sistema de control de versiones distribuido bajo licencia de Software libre: *Subversion*.
- Un procedimiento de informe de errores denominado *Asterisk Bug Tracker*. Este procedimiento además de para reportar errores y hacer el seguimiento de sus soluciones, sirve a su vez para mantener un sistema de méritos, Karma, que contiene un ranking de colaboradores con una puntuación de acuerdo a las aportaciones que éstos han realizado.

La estructura organizativa que permite el desarrollo de Asterisk consta de los siguientes elementos:

- Organizador y principal desarrollador: Mark Spencer
- Administradores: grupo de colaboradores que apoyan a Spencer. Los administradores realizan principalmente labores de programación y control del software generado.
- Managers: amplio grupo de programadores, que aportan soluciones a errores documentados y crean nuevas funcionalidades.
- Reporters, todos aquellos colaboradores que realizan informes sobre errores detectados.

Toda nueva funcionalidad es probada exhaustivamente antes de formar parte del repositorio del sistema de control de versiones y ha de contar finalmente con el visto bueno de los responsables de los repositorios, de acuerdo con criterios de oportunidad, prioridad o importancia de la nueva funcionalidad propuesta.

El modelo de negocio basado en la Licencia de Software Libre facilita por otro lado la existencia de numerosas empresas relacionadas con el programa Asterisk que aportan valor añadido al software mediante el diseño, instalación y mantenimiento de centralitas de telefonía basadas en Asterisk, así como la formación de usuarios, administradores y desarrolladores de centralitas basadas en Asterisk.

Digium, la empresa que como hemos comentado se encarga del desarrollo oficial de Asterisk, amplía este modelo de negocio por dos vías. La primera vía es la venta de hardware específico, principalmente tarjetas telefónicas que se añaden a los PCs y permiten interconectar las centralitas telefónicas Asterisk con la Red Telefónica Conmutada tradicional. La segunda línea de negocio se encarga de la venta de software propietario entre el cual destaca la Asterisk Business Edition, que añade funcionalidades que contienen Copyright.

2.3.2. Características de Asterisk

Las soluciones basadas en Asterisk poseen ventajas en cuanto a coste y flexibilidad de configuración frente a las centralitas telefónicas tradicionales. Además, este software cuenta con desarrollos ya implementados de muchas funcionalidades que anteriormente sólo estaban disponibles en los costosos sistemas propietarios de PBX, tales como las que se enumerarán a continuación. Además los usuarios pueden crear nuevas funcionalidades de una manera sencilla escribiendo un *dialplan* en el lenguaje de script de Asterisk o de modo más complejo añadiendo módulos escritos en lenguaje C o en cualquier otro lenguaje de programación reconocido por Linux.

Las principales funcionalidades de Asterisk son, entre otras:

- Completo sistema de buzón de voz, incluyendo el envío de los mensajes a direcciones de correo electrónico
- Desvío de llamadas (automático, si ocupado o si no disponible)
- Transferencia de llamadas a otros usuarios de la centralita (transferencia directa y transferencia con consulta)
- Llamadas en espera
- Identificación de llamadas
- Captura de llamadas
- Multiconferencia
- Sistema de autenticación de usuarios
- Listas negras de usuarios
- Monitorización de llamadas
- Registros de detalles de llamada, útiles para módulos de tarificación
- Recuperación de llamadas perdidas
- Grabación de llamadas
- Servicio de música en espera
- Recepción y envío de fax
- Sistema de autenticación de usuarios
- Mensajería SMS
- Posibilidad de utilización como pasarela a otras centralitas
- Conversión entre formatos de datos (*transcoding*)
- Conversión entre protocolos
- Integración con bases de datos relacionales, servicios Web y LDAP

La configuración y personalización del funcionamiento de una centralita basada en Asterisk se basa principalmente en unos ficheros de texto editables. Dichos ficheros, denominados “ficheros de configuración” se componen de una sintaxis particular del sistema Asterisk, el denominado anteriormente lenguaje script de Asterisk. El conocimiento avanzado de dicha sintaxis hace posible implementar tanto el *dialplan* como todas las funcionalidades mencionadas deseadas. Por ello se suele decir que Asterisk es una caja de herramientas o una plataforma de desarrollo, ya que contiene los elementos necesarios para construir las soluciones de comunicaciones citadas anteriormente.

Para conectar teléfonos estándar analógicos a una centralita software Asterisk es necesario instalar tarjetas electrónicas telefónicas FXS o FXO. Estas tarjetas son fabricadas por Digium y por otros proveedores, y sin ellas no se podría conectar el servidor Asterisk a la red telefónica convencional, ya que un módem no es suficiente.

Otra característica interesante de Asterisk es que proporciona herramientas que dan soporte a la transmisión de voz tanto por líneas telefónicas convencionales, como por redes de datos. Por un lado gestiona y transmite la voz por los canales tradicionales tales como líneas analógicas, líneas RSDI (ISDN/BRI) y enlaces digitales T1/E1. Por otro lado reconoce muchos protocolos VoIP como pueden ser SIP, H.323, IAX y MGCP. Asterisk puede interoperar con terminales IP actuando como un registrador y como pasarela o *gateway* entre ambos. Soporta los tipos de señalización estándar europeo y americano, así como otros codecs utilizados en los sistemas de telefonía empresariales.

Según datos de Digium, hoy en día más de dos millones de usuarios utilizan soluciones de telefonía basadas en Asterisk®, haciendo de éste el proyecto líder de código abierto sobre telefonía y un factor clave en el crecimiento de la VoIP.

2.3.3. Organización y versiones de Asterisk

El desarrollo del código de Asterisk se organiza en módulos, y esos módulos evolucionan siguiendo unas versiones.

La versión estable de Asterisk está compuesta por los siguientes módulos:

- Asterisk: Ficheros base del proyecto.
- *Sounds*: Aporta sonidos y frases en diferentes idiomas. (Incluidos en el paquete Asterisk)
- DAHDI: Soporte para hardware. Drivers de tarjetas. (Anteriormente ZAPTEL)
- *Addons*: Complementos y añadidos del paquete Asterisk. Opcional.
- *Libpri*: Soporte para conexiones digitales. Opcional.

Cada módulo cuenta con una versión estable y una versión de desarrollo. La forma de identificar las versiones se realiza mediante la utilización de tres números separados por un

punto. El primer número ha sido desde el inicio del desarrollo de Asterisk el uno, el segundo número indica la versión, mientras que el tercero muestra la revisión liberada. En las revisiones se llevan a cabo correcciones, pero no se incluyen nuevas funcionalidades.

En las versiones de desarrollo el tercer valor siempre es un cero, seguido de la palabra "beta" y un número, para indicar la revisión.

A continuación podemos ver una tabla que muestra el histórico de versiones de Asterisk que continúan en activo hasta la fecha de realización de esta memoria:

Serie de versiones	**Tipo de la versión**	**Fecha de la versión**	**Únicamente parches de seguridad**	**Fin de vida esperado**
1.2.X		2005-11-21	2007-08-07	2010-11-21
1.4.X	LTS	2006-12-23	2010-12-23	2011-12-23
1.6.0.X	Standard	2008-10-01	2010-04-01	2010-10-01
1.6.1.X	Standard	2009-04-27	2010-04-27	2011-04-27
1.6.2.X	Standard	2009-12-18	2010-12-18	2011-12-28
1.8.X	LTS	TBD (Objetivo: Q2 2010)	TBD + 4 años	TDB + 5 años

El tipo de versión define el tiempo que será mantenida. Una versión de soporte a largo plazo (indicado por sus siglas en inglés, *Long Term Support*) tendrá un mantenimiento completo durante 4 años, con un año adicional para parches de seguridad. Una versión estándar es mantenida un periodo de tiempo más corto, que será un mínimo de un año de mantenimiento completo más un año de mantenimiento adicional para parches de seguridad.

Como se puede ver en la tabla superior, la versión estable con mantenimiento a largo plazo a fecha de comienzo de realización del proyecto en noviembre de 2009 era la 1.4, que es la versión de Asterisk utilizada por la centralita telefónica existente IRI.

No obstante para este proyecto se realizó un desarrollo paralelo en una “rama” separada (como podremos ver más adelante en el apartado que explica el software *Maven*) que utiliza la versión estándar de Asterisk 1.6.1. La nueva versión fue utilizada en lugar de la versión existente debido a que entre las nuevas funcionalidades ofrecidas se encuentra el soporte en tiempo real de bases de datos LDAP mediante el nuevo módulo “res_config_ldap” y un nuevo módulo de esquema LDAP, características fundamentales para este proyecto.

Como la versión 1.6 no tiene soporte a largo plazo, no está recomendado utilizarla en sistemas en producción, por lo que habría que esperar al lanzamiento de la próxima versión 1.8 que integre la nueva funcionalidad. Este lanzamiento está previsto para el segundo trimestre de 2010.

2.3.4. Asterisk en LDAP

Desde mediados de 2008 se introdujo en la versión beta de Asterisk 1.6.x el módulo "res_config_ldap.c". Este módulo permite la utilización de LDAP con Asterisk en tiempo real.

Tras la instalación de la versión actualizada de Asterisk que contiene el nuevo módulo, hay que actualizar algunos ficheros de configuración tanto en el lado del servidor Asterisk como en el lado del directorio LDAP.

En Asterisk, hay que configurar ficheros como *extconfig*, *res_ldap.conf* y *extensions.conf*. El fichero *res_ldap.conf* contiene un mapeado de los valores de tiempo real de Asterisk a objetos LDAP.

En LDAP, la definición de los atributos y las *ObjectClasses* para dar soporte a usuarios de Asterisk vienen definidos en el esquema de LDAP *Asterix.schema*, que debe ser añadido a los esquemas del directorio LDAP que se vaya a integrar con Asterisk.

Asterix.schema contiene todos los atributos necesarios para Asterisk. De esta forma, cualquiera de los atributos definidos en dicho esquema puede ser asociado a objetos del directorio que implementen la clase *ObjectClassAsteriskSIPUser* (ver detallado el *Asterix.schema* en los anexos en este documento).

En concreto los atributos utilizados en este proyecto han sido:

```
AstContext
AstExtension
AstAccountCallerID
AstAccountContext
AstAccountMailbox
AstAccountQualify
AstAccountType
AstAccountDisallowedCodec
AstAccountCanReinvite
AstAccountHost
AstAccountNAT
AstAccountAllowedCodec
AstAccountSecret
AstAccountName
```

Continuaremos con una descripción más detallada de los pasos a seguir para la configuración de todos los ficheros implicados en la integración en el anexo dedicado a la instalación y configuración de la aplicación.

2.4. Protocolos de VoIP

Una conexión VoIP implica principalmente dos procesos:

- una serie de transacciones de señalización entre los puntos finales y entre los nodos intermedios
- los flujos de datos bidireccionales, es decir, el contenido de la conversación.

El conjunto de reglas que rigen estos procesos son los protocolos de comunicación. Para el caso de la VoIP existen múltiples protocolos, de los cuales veremos los principales a continuación.

2.4.1. Protocolo SIP

SIP (*Session Initiation Protocol*) es un protocolo de señalización cuya función principal es crear, modificar y terminar sesiones a través de redes IP. Permite localizar a los usuarios e intercambiar información de los medios implicados en la sesión, a la vez que es totalmente independiente del tipo de sesión a establecer, por lo que puede ser usado para iniciar conversaciones de voz, videoconferencias, aplicaciones compartidas, etc.

El protocolo SIP fue estandarizado por la IETF (*Internet Engineering Task Force*) mediante la RFC 2543 en su versión 1.0, realizando una nueva versión 2.0 recogida en la RFC 3261. En esta última RFC se describe el funcionamiento básico del protocolo, citando para algunos aspectos concretos otras RFCs como son la RFC 3262 y RFC 3265. Además, en el documento original está prevista la ampliación del protocolo mediante nuevas RFCs, lo que ha permitido la aparición de numerosas extensiones que van aumentando las funcionalidades y posibilidades del protocolo.

En un principio, SIP no tuvo mucha acogida, siendo el protocolo H.323 el protocolo de señalización mayoritario en VoIP. Sin embargo, en los últimos años ha ganado gran popularidad debido entre otros factores al hecho de que su especificación está libremente disponible y a que es un protocolo relativamente simple, basado en mensajes de petición y respuesta, reutilizando sintaxis de protocolos muy conocidos como HTTP y SMTP.

SIP es un protocolo de señalización a nivel de aplicación diseñado para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervengan elementos multimedia, como video o VoIP. Utiliza el puerto 5060 de los protocolos de transporte TCP y UDP y aunque está diseñado para trabajar en VoIP sobre ambos, la versión actual de Asterisk tiene soporte de SIP sobre UDP.

En este punto se nos plantea la pregunta de por qué SIP emplea UDP en vez de TCP para transmitir voz sobre IP. La respuesta es que TCP es un protocolo orientado a conexión, por lo que si se pierde un paquete, éste se retransmite y puede que incluso se descarte la ventana de paquetes si el reenvío tarda demasiado. Esto provoca que en la capa de aplicación no se observe pérdida de paquetes, pero a cambio se detectarían saltos en la voz. UDP, por su parte, no está orientado a conexión, por lo que si un paquete se pierde no ocurre nada. Por el mecanismo de comprensión de los codecs actuales, en caso de pérdida

de un paquete la voz sigue resultando comprensible. Esto es importante si analizamos la mecánica del protocolo UDP. Dicha mecánica consiste en mantener una cola de paquetes no reproducidos y ordenarlos según su hipotético tiempo de reproducción. Hasta aquí es similar a TCP. Sin embargo, la política de UDP es reproducir el primer paquete de la cola cuando le toque por su tiempo marcado, independientemente de que falten paquetes intermedios entre el último reproducido y el primero de la cola. Como consecuencia, en caso de que haya un problema de red puntual, el codec empleado puede tratar de recuperar, por distintas técnicas, los fragmentos de voz intermedios. La voz sonará extraña o "metalizada" o, a lo sumo, se perderá una pequeña fracción de segundo de reproducción. Todo esto no es posible con TCP ya que, en caso de existir problemas en la red, se esperaría a tener todos los fragmentos de voz. Esto provocaría que, pese a tener la voz con completa calidad, la conversación no sería operativa debido a los tiempos de espera. Con UDP se puede degradar la voz, pero no se perderá la dinámica de la conversación.

Conviene destacar que SIP únicamente se dedica a la señalización. El transporte del tráfico de voz se realiza mediante *streams* del protocolo RTP (*Real Time Protocol*).

SIP es un protocolo extremo a extremo, lo que implica que toda la lógica se encuentra almacenada en los dispositivos finales (salvo el rutado de los mensajes). El estado de la conexión también se encuentra en los dispositivos finales. Este tipo de arquitectura facilita una gran escalabilidad, ya que no hay que tocar los elementos intermedios de la red. Sin embargo, el precio a pagar es la existencia de cabeceras de gran tamaño, ya que al encontrarse la lógica en los sistemas finales hay que mandar gran cantidad de información de control entre los mismos.

El protocolo SIP realiza un intercambio de mensajes de forma textual. Los mensajes SIP pueden ser de dos tipos:

- peticiones o métodos (*requests*), y
- respuestas o códigos de estados (*responses*),

Estos mensajes están compuestos siguiendo la RFC 2822, que consiste en una línea inicial seguida de uno o más campos de cabecera (*headers*), una línea vacía que indica el final de las cabeceras, y por último el cuerpo del mensaje, siendo éste un campo opcional.

Una transacción SIP está formada por una petición y una o varias respuestas.

La primera línea de la cabecera indica el tipo: petición o respuesta.

A continuación, las cinco cabeceras *Via, From, To, Call-ID*, y *Cseq* identifican de forma unívoca cada mensaje dentro cada establecimiento de sesión. Su significado es el siguiente:

- *Via*: Indica el transporte usado para el envío e identifica la ruta del *request*, por ello cada *Proxy* añade una línea a este campo.
- *From*: Indica la dirección del origen de la petición.
- *To*: Indica la dirección del destinatario de la petición.
- *Call-Id*: Identificador único para cada llamada. Contiene la dirección del *Host*. Debe ser igual para todos los mensajes dentro de una transacción. En los

mensajes de tipo OPTIONS, el campo *Call-ID* permite relacionar las peticiones y respuestas.

- *Cseq*: Se inicia con un número aleatorio e identifica de forma secuencial cada petición.

Los métodos definidos en la norma básica del protocolo SIP son **INVITE, ACK, CANCEL, BYE, REGISTER y OPTIONS**. Su significado es el siguiente:

- **INVITE**: Permite invitar un usuario o servicio a participar en una sesión, o bien a modificar parámetros en una sesión ya existente.
- **ACK**: Confirma el establecimiento de una sesión.
- **OPTION**: Solicita información sobre las capacidades de un servidor.
- **BYE**: Indica la terminación de una sesión.
- **CANCEL**: Cancela una petición pendiente.
- **REGISTER**: Registrar al Agente de Usuario *(User Agent,* entidad SIP).

En las nuevas revisiones del protocolo se añaden nuevos métodos, pensados para realizar control de presencia, mensajería instantánea, transferencias de sesión etc. Algunos de estos métodos son **SUBSCRIBE, NOTIFY, MESSAGE, UPDATE, etc.**

El formato de los mensajes de respuesta, es similar al anterior, difiriendo en la línea inicial, llamada *Status-Line*, que contiene la versión de SIP, el código de la respuesta (*Status–Code*) y una pequeña descripción (*Reason-Phrase*). El código de respuesta esta compuesto por 3 dígitos, coincidiendo con los usados en HTTP, y que dependiendo del primer dígito del código tiene un significado distinto. Se pueden dividir en respuestas provisionales (1xx) y finales (2xx-6xx). Los tipos de respuesta en función del primer dígito son:

- **1xx:** respuestas provisionales, dan información pero no finalizan una transacción.
- **2xx:** respuestas de éxito, indican que la petición ha tenido éxito.
- **3xx:** respuestas de redirección, la petición no ha tenido éxito pero redirigen a otra dirección.
- **4xx:** respuestas de error del cliente que hizo la petición.
- **5xx:** respuestas de error del servidor que atiende la petición.
- **6xx**: respuestas de error globales a toda la red.

A continuación describiremos las entidades SIP que pueden intervenir en una sesión.

- **UA (User Agent)**: Un usuario SIP necesita de un Agente de Usuario (UA), para enviar y recibir mensajes SIP. Podemos encontrar dos tipos de agentes: Agente de Usuario de Cliente (UAC) que crea peticiones SIP y Agente de

Usuario Servidor (UAS) que interactúa con el usuario recibiendo una petición SIP y contestado a ella.

- **Registrar:** Es la entidad que recibe peticiones de tipo REGISTER. Estas peticiones enviadas por los UAs para asociar su dirección SIP pública conocida (usuario@dominio.es) a la dirección SIP actual concreta (usuario@X.X.X.X:XXXX). El Registrar actualiza una base de datos de usuarios registrados llamada *Location Server*.
- **Location Server**: Esta entidad es una base de datos con información de los usuarios registrados. Es una entidad lógica que no se comunica mediante SIP. Se puede implementar como una base de datos en memoria, una base de datos externa accesible mediante lenguaje SQL, LDAP, etc. Es actualizada por el Registrar y consultada por un Servidor SIP para encaminar los mensajes a la localización actual de los usuarios.
- **Proxy Server:** Es el Servidor SIP más habitual. Recibe una petición SIP y la reenvía a la entidad destinataria. Para ello consulta el *Location Server*, o la envía según una conducta programada que puede depender del usuario llamado, la hora del día, etc. Puede mantener distintos niveles de contexto según sea *stateless* (no mantiene información entre mensajes), *stateful* (implementa máquina de estados para cada transacción) o *call stateful* (se mantiene en el camino de señalización durante toda la sesión).
- **Redirect Server**: Es el Servidor SIP más simple. En vez de reenviar los mensajes a la entidad adecuada siempre responde con una respuesta especial de redirección, indicando la dirección adecuada a la que hay que enviar la petición. El UA que recibe este mensaje de redirección debe repetir la petición enviándola a la URL a la que ha sido redirigido.
- **Back-to-Back User Agent (B2BUA)**: Es también un Servidor SIP, formada por dos UA. Uno recibe la llamada como si fuera para él, e inicia otra sesión con el destino mediante otra instancia de UA. Desde el punto de vista de la señalización, ambos extremos de la llamada ven como interlocutor al B2BUA que es el que se encarga de generar en una sesión la señalización recibida en la otra sesión. Se podría pensar en su funcionamiento como una pasarela SIP-SIP. Esta entidad permite un mayor control sobre las sesiones ya que a diferencia de los servidores comentados, el B2B puede generar espontáneamente peticiones SIP hacia los dos extremos permitiendo, por ejemplo, terminar una sesión.

2.4.1.1. La identificación de usuarios en el direccionamiento SIP.

Las entidades SIP identifican a un usuario con las SIP URI (*Uniform Resource Identifier*). Una SIP URI tiene un formato similar al de una dirección de una dirección de correo electrónico, componiéndose de un nombre de usuario y un dominio delimitado por una @, como muestran los siguientes casos:

- usuario@dominio, donde dominio es un nombre de dominio completo.
- usuario@equipo, donde equipo es el nombre de la máquina.
- usuario@dirección_ip, donde dirección_ip es la dirección IP del dispositivo.
- número_teléfono@gateway, donde el *gateway* permite acceder al número de teléfono de la Red Telefónica Conmutada.

Al igual que ocurre en el caso de redes como Internet, en SIP existen también unos procedimientos DNS utilizados por los clientes para traducir una SIP URI en una dirección IP, puerta y protocolo de transporte utilizado, o por los servidores para retornar una respuesta al cliente en caso de que la petición falle.

2.4.1.2. Ejemplo de protocolo SIP

Una vez examinados distintos aspectos técnicos del protocolo SIP vamos a presentar un ejemplo de los siguientes pasos que se suceden en el transcurso de una llamada.

Dichos pasos se pueden representar gráficamente de la siguiente manera:

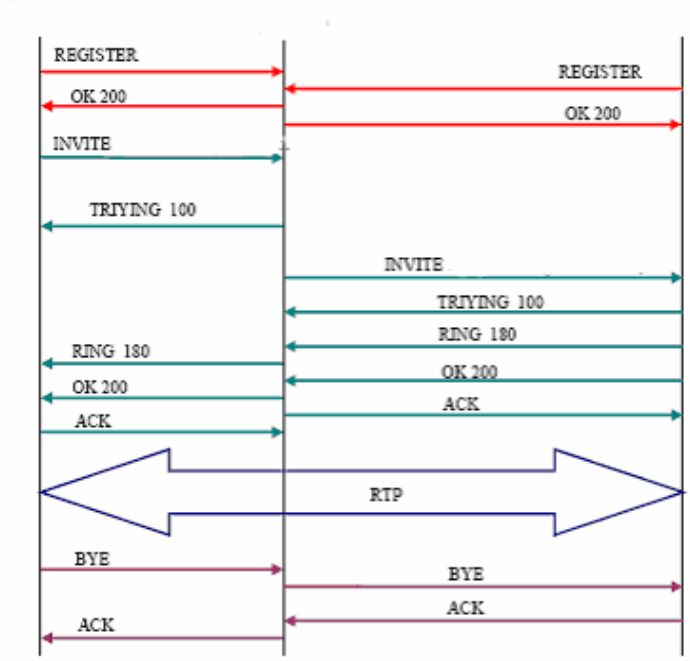


Figura 4. Ejemplo protocolo SIP. Fuente: VoIPforo

Los extremos de la conexión simbolizan a los usuarios A y B, también llamados los "terminales" de la comunicación, y la línea intermedia el servidor *proxy*.

Las dos primeras transacciones corresponden al registro de los usuarios, ya que éstos deben registrarse para poder ser encontrados por otros usuarios. En este caso, los terminales envían una petición REGISTER, donde los campos 'from' y 'to' de la cabecera corresponden al usuario registrado. El servidor *proxy*, que actúa como *Register*, consulta si el usuario puede ser autenticado y envía un mensaje de OK en caso positivo.

La siguiente transacción corresponde a un establecimiento de sesión. Esta sesión consiste en una petición INVITE del usuario al *proxy*. Inmediatamente, el *proxy* envía un TRYING 100 para parar las retransmisiones y reenvía la petición al usuario B. El usuario B envía un RING 180 cuando el teléfono empieza a sonar y también es reenviado por el *proxy* hacia el usuario A.

Por ultimo, el OK 200 corresponde a aceptar la llamada (el usuario B descuelga).

En este momento la llamada está establecida, por lo que pasa a funcionar el protocolo de transporte RTP con los parámetros establecidos en la negociación, tales como puertos, direcciones, codecs, etc. Dicha negociación ha sido previamente realizada mediante otro protocolo llamado SDP, protocolo de descripción de sesión.

La última transacción corresponde a una finalización de sesión. Esta finalización se lleva a cabo con una única petición BYE enviada al Proxy, y posteriormente reenviada al usuario B. Este usuario contesta con un OK 200 para confirmar que se ha recibido el mensaje final correctamente.

2.4.1.3. SIP y NAT

Uno de los mayores obstáculos técnicos a los que se enfrenta el protocolo SIP es el reto de realizar correctamente todas las transacciones cuando NAT está presente.

NAT (Traducción de Direcciones de Red) es un estándar creado por el IETF para traducir un rango de direcciones privadas en una dirección pública.

Las direcciones privadas son rangos especiales de direcciones IP que se reservan para ser utilizadas en redes locales, y se llaman "privadas" o "no enrutables" porque no pueden ser utilizadas en Internet. Los routers intermedios que componen Internet no entienden este tipo de direcciones y no las encaminan. Esto da gran flexibilidad ya que pueden utilizarse ese tipo de direcciones privadas libremente dentro de cada red local, sin ningún tipo de trámite. Sin embargo, con esas direcciones no se puede enviar información al exterior ya que en las cabeceras de los paquetes IP figura la dirección y, como se ha comentado, los routers intermedios no entienden esas direcciones. NAT es un estándar que está diseñado para solucionar este problema.

La idea básica que hay detrás de NAT es traducir las IPs privadas de la red en una IP pública para que la red pueda enviar paquetes al exterior; y traducir luego esa IP pública, de nuevo a la IP privada del ordenador que envió el paquete, para que pueda recibirlo una vez llega la respuesta.

Sin embargo existen protocolos que no funcionan del todo correctamente cuando NAT está presente y SIP es uno de ellos. Los paquetes SIP enviados desde un ordenador que se encuentra bajo NAT tienen en su cabecera como dirección origen la dirección IP privada, y por lo tanto no enrutable. Estos paquetes no son procesados por NAT, que actúa en un nivel inferior. Por tanto, la información de direccionamiento no se modifica y, de este modo, los datos de la conversación, llevada sobre RTP, no llevarán la información de dirección correcta. Además de esto, muchos *firewalls* que llevan NAT integrado no consideran los datos de la conversación como parte de la transacción SIP y, por tanto, si bien las operaciones de señalización se realizan correctamente no ocurre lo mismo con el envío de los datos de la conversación, siendo necesario abrir en el firewall una serie de puertos destinados a los paquetes RTP.

No obstante, SIP ha sido diseñado teniendo en cuenta a NAT, por lo que aunque de manera general sí puede presentar algunos problemas de configuración, cada aplicación particular que utilice SIP puede tener sus propias soluciones específicas para solucionar los problemas con NAT, como ocurre con Asterisk. Aún así podemos esperar que futuras versiones de SIP mejoren este problema.

2.4.2. Comparativa de otros protocolos de VoIP con SIP

2.4.2.1. H.323

El protocolo H.323 fue diseñado por la ITU (*Internacional Telecommunications Union*) con el objetivo de proveer un mecanismo de transporte sobre IP para videoconferencia, pero evolucionó rápidamente para convertirse en un protocolo con gran fama para emplearlo en VoIP.

Un punto fuerte de H.323 fue la relativa y temprana disponibilidad de un grupo de estándares, no solo definiendo el modelo básico de llamada, sino que además definía servicios suplementarios, necesarios para dirigir las expectativas de comunicaciones comerciales. H.323 fue el primer estándar de VoIP en adoptar el estándar RTP para transportar audio y vídeo sobre redes IP.

H.323 vs SIP

H.323 ha sido tradicionalmente el protocolo escogido para la señalización en el entorno VoIP. Sin embargo, ese dominio se ha reducido significativamente en los últimos años en favor del protocolo SIP. Si tomamos como referencia los medios de comunicación especializados, H.323 se menciona con menos regularidad que SIP, pese a que es un protocolo que en algunos aspectos es técnicamente superior.

Uno de los factores que hacen que H.323 no sea tan popular como SIP es su complejidad. H.323 no es un protocolo como tal, sino una pila compuesta por muchos protocolos específicos, que realizan una gran cantidad de operaciones. Esto hace que las implementaciones de H.323 sean más complejas y difíciles de mantener. Por el contrario, SIP sigue una filosofía más práctica y simple reaprovechando la estructura y la sintaxis de otros protocolos muy conocidos y que lo hacen fácil de implementar, depurar y optimizar.

Otra desventaja de H.323 frente a SIP es su comportamiento frente a NAT. Es cierto que, a diferencia de otros protocolos como IAX o Skype que se comportan perfectamente frente a NAT, tanto SIP como H.323 adolecen de problemas frente a esta tecnología. Sin embargo, SIP dispone de una serie de soluciones ad-hoc, específicas para cada caso, que pueden permitir solventar el problema de NAT con mayor o menor éxito. Por el contrario, H.323 se comporta mal frente a NAT, y las posibles soluciones son bastante complejas.

H.323 aún lleva la mayor parte del tráfico mundial de VoIP. Esto es así ya que ha sido el protocolo escogido tradicionalmente por los grandes operadores de VoIP. Sin embargo, el mundo de las telecomunicaciones se hace cada vez más abierto, y hay menos dependencia de los operadores tradicionales, permitiendo que soluciones menos complejas toman más protagonismo, como es el caso de SIP.

“VoIP foro” proporciona la siguiente tabla comparativa entre SIP y H.323, cubriendo los aspectos de arquitectura, funcionalidad básica y avanzada, escalabilidad y seguridad entre otras (fecha de publicación: Septiembre de 2009):

	H.323	**SIP**
Arquitectura	H.323 cubre casi todos los servicios como capacidad de intercambio, control de conferencia, señalización básica, calidad de servicio, registro, servicio de descubrimiento y más.	SIP es modular y cubre la señalización básica, la localización de usuarios y el registro. Otras características se implementan en protocolos separados.
Componentes	Terminal/Gateway	UA
	Gatekeeper	Servidores
Protocolos	RAS/Q.931	SIP
	H.245	SDP
Funcionalidades de control de llamada		
Transferencia de llamada (Call Transfer)	Si	Si
Expedición de llamada (Call Forwarding)	Si	Si
Tenencia de llamada (Call Holding)	Si	Si
Llamada estacionada/recogida (Call Parking/Pickup)	Si	Si
LLamada en espera (Call Waiting)	Si	Si
Indicación de mensaje en espera (Message Waiting Indication)	Si	No
Identificación de nombre (Name Identification)	Si	No
Terminación de llamada con subscriptor ocupado (Call Completion on Busy Subscriber)	Si	Si
Ofrecimiento de llamada (Call Offer)	Si	No
Intrusión de llamada (Call Intrusion)	Si	No
	H.323 las divide en los protocolos H.450, RAS, H.245 y Q.931	

Características Avanzadas		
Señalización multicast (Multicast Signaling)	Si, requiere localización (LRQ) y descubrimiento automático del gatekeeper (GRQ).	Si, ejemplo, a través de mensajes de grupo INVITEs.
Control de la llamada de un tercero (Third-party Call Control)	Si, a través de pausa de la tercera parte y re-enrutando según esta definido en H.323. Un control más sofisticado se define en el estándar de las series H.450.x.	Si, según se describe en los borradores (Drafts) del protocolo.
Conferencia	Si	Si
Pinchar para llamar (Click for Dial)	Si	Si
Escalabilidad		
Número amplio de dominios (Large Number of Domains)	La intención inicial de H.323 fue el soporte de LANs, por lo que está pensado para el direccionamiento de redes amplias. El concepto de zona fue añadido para acomodar este direccionamiento amplio. Los procedimientos son definidos por localización de usuarios a través de nombres de email. Se define la comunicación entre dominios administrativos, describiendo los métodos para resolución de direcciones, autorización de acceso y el reporte entre dominios administrativos. En las búsquedas multidominio no hay formas sencillas de detectar bucles. La detección de bucles se puede realizar a través del campo "PathValue" pero introduce problemas relativos a la escalabilidad.	SIP soporta de manera inherente direccionamientos de áreas. Cuando muchos servidores están implicados en una llamada SIP usa un algoritmo similar a BGP que puede ser usado en una manera sin estado evitando problemas de escalabilidad. Los SIP Registrar y servidores de redirección fueron diseñados para soportar localización de usuarios.
Gran cantidad de llamadas (Large Number of Calls)	El control de llamadas en se implementa de una manera sin estado. Un gateway usa los mensajes definidos en H.225 para ayudar al gatekeeper en el balanceo de carga de los gateways implicados.	El control de llamadas en se implementa de una manera sin estado. SIP soporta escalabilidad n a n entre UAs y servidores. SIP necesita menos ciclos de CPU para generar mensajes de señalización Por lo tanto, teóricamente un servidor puede manejar más transacciones. SIP ha especificado un método de balanceado de carga basado en el mecanismo de traslación DNS SRV.

Estado de la conexión	Con estado o sin estado.	Con estado o sin estado. Una llamada SIP es independiente de la existencia de una conexión en la capa de transporte, pero sin embargo la señalización de llamadas tiene que ser terminada explícitamente.
Internacionalización	Si, H.323 usa Unicode (BMPString con ASN.1) para alguna información textual (h323-id), pero generalmente tiene pocos parámetros textuales	Si, SIP usa Unicode (ISO 10646-1), codificado como UTF-8, para todas las cadenas de texto, permitiendo todos los caracteres para nombres, mensajes y parámetros. SIP provee métodos para la indicación del idioma y preferencias del idioma.
Seguridad	Define mecanismos de seguridad y facilidades de negociación mediante H.235, puede usar SSL para seguridad en la capa de transporte.	SIP soporta autentificación de llamante y llamado mediante mecanismos HTTP. Autenticación criptográfica y encriptación son soportados salto a salto por SSL/TSL pero SIP puede usar cualquier capa de transporte o cualquier mecanismo de seguridad de HTTP, como SSH o S-HTTP. Claves para encriptación multimedia se ofrecen usando SDP. SSL soporta autenticación simétrica y asimétrica. SIP también define autenticación y encriptación final usando PGP o S/MIME.
Interoperabilidad entre versiones	La compatibilidad hacia atrás de H.323 permite que todas las implementaciones basadas en diferentes versiones de H.323 sean fácilmente integrables.	En SIP, una nueva versión puede descartar características que no van a ser soportadas más. Esto consigue reducir el tamaño del código y la complejidad del protocolo, pero hace perder cierta compatibilidad entre versiones.
Implementación de la Interoperabilidad	H.323 provee una guía de implementación, que clarifica el estándar y ayuda a la interoperabilidad entre diferentes implementaciones.	SIP no prevé ninguna guía de interoperabilidad
Facturación	Incluso con el modelo de llamada directa H.323, la posibilidad de facturar la llamada no se pierde porque los puntos finales reportan al gatekeeper el tiempo de inicio y finalización de la llamada mediante el protocolo RAS.	Si un proxy SIP quiere recoger información de facturación no tiene otra opción que revisar el canal de señalización de manera constante para detectar cuando se completa la llamada. Incluso así, las estadísticas están sesgadas porque la señalización de la llamada puede tener retardos.

Codecs	H.323 soporta cualquier codec, estandarizado o propietario, no sólo codecs ITU-T, por ejemplo codecs MPEG o GSM. Muchos fabricantes soportan codecs propietarios a través de ASN.1 que es equivalente en SIP a "códigos privados de mutuo acuerdo" Cualquier codec puede ser señalizado a través de la característica GenericCapability añadida en H.323v3.	SIP soporta cualquier codec IANA-registered (es una característica heredada) o cualquier codec cuyo nombre sea de mutuo acuerdo.
Bifurcación de llamadas (Call Forking)	Un gatekeeper H.323 puede controlar la señalización de la llamada y puede bifurcar a cualquier número de dispositivos simultáneamente.	Un proxy SIP puede controlar la señalización de la llamada y puede bifurcar a cualquier número de dispositivos simultáneamente.
Protocolo de transporte	Fiable (Reliable) o no fiable (unreliable), Ej., TCP o UDP. La mayoría de las entidades H.323 usan transporte fiable (TCP) para señalización.	Fiable (Reliable) o no fiable (unreliable), Ej., TCP o UDP. La mayoría de las entidades SIP usan transporte no fiable (UDP) para señalización.
Codificación de mensajes (Message Encoding)	H.323 codifica los mensajes en un formato binario compacto adecuado para conexiones de gran ancho de banda.	SIP codifica los mensajes en formato ASCII, adecuado para que lo puedan leer los humanos.
Direccionamiento (Addressing)	Mecanismos de señalización flexibles, incluyendo URLs y números E.164.	SIP sólo entiende direcciones del estilo URL.
Interconexión Red Telefónica Pública (PSTN Interworking)	H.323 toma prestado de la red telefónica pública protocolos como Q.931 y está por tanto bien adecuada para la integración. Sin embargo, H.323 no emplea la analogía a tecnología de conmutación de circuitos de red telefónica pública de SIP. H.323 es totalmente una red de conmutación de paquetes. El como los controles deben implementarse en la arquitectura H.323 está bien recogido en el estándar.	SIP no tiene nada en común con la red telefónica pública y esa señalización debe ser "simulada" en SIP. SIP no tiene ninguna arquitectura que describa cómo deben implementarse los controles.
Detección de bucles (Loop Detection)	Si, los gatekeepers pueden detectar bucles mirando los campos "CallIdentifier" y "destinationAddress" en los mensajes de procesamiento de la llamada. Combinando ambos se pueden detectar bucles	Si, el campo "Via" de la cabecera de los mensajes SIP facilita el proceso. Sin embargo, este campo "Via" puede generar complejidad en los algoritmos de detección de bucles y se prefiere usar la cabecera "Max-Forwards" para limitar el número de saltos y por tanto los

		bucles.
Puertos mínimos para una llamada VoIP	5 (Señalización de llamada, 2 RTP, y 2 RTCP.)	5 (Señalización de llamada, 2 RTP, y 2 RTCP.)
Conferencias de vídeo y datos	H.323 suporta todo tipo de conferencia de vídeo y datos. Los procedimientos permiten control de la conferencia y sincronización de los streams de audio y vídeo,	SIP no soporta protocolos de vídeo como T.120 y no tiene ningún protocolo para control de la conferencia.

2.4.2.2. IAX

El protocolo IAX (*Inter-Asterisk Exchange Protocol*) es un protocolo creado por Digium, la empresa que realiza el desarrollo oficial de Asterisk, con el objetivo de comunicar distintos servidores Asterisk entre sí. Es un protocolo abierto, lo que permite que cualquiera pueda acceder a él libremente y contribuir a su desarrollo, pero no constituye estándar.

La versión actual del protocolo es la segunda (IAX2) ya que la primera está obsoleta. Por ello, se sobreentiende que se está hablando de la segunda versión del protocolo independientemente si se le nombra como IAX o IAX2.

IAX utiliza un único puerto UDP, generalmente el 4569, tanto para el transporte de la información de señalización como para el transporte de los *streams* RTP, los cuales contienen el tráfico de voz.

Una virtud del protocolo IAX es que soporta *trunking*. De este modo, los datos de múltiples llamadas pueden ser manejados en un único conjunto de paquetes, lo que significa que un datagrama IP puede entregar información para más llamadas sin crear latencia adicional. Esto es una gran ventaja para los usuarios de VoIP, donde las cabeceras IP constituyen un gran porcentaje del ancho de banda utilizado.

IAX permite la autenticación mediante tres métodos: texto plano, *hash* MD5 e intercambio de claves RSA.

La versión 2 del protocolo fue específicamente diseñada para poder trabajar detrás de dispositivos que incorporen NAT. Éste es uno de sus principales puntos fuertes de cara a ser un protocolo utilizado ampliamente en el futuro.

Igualmente, el hecho de utilizar un único puerto tanto para la señalización como para la transmisión de contenido hace que el número de posibles vulnerabilidades tras un firewall se reduzca al mínimo. Estas consideraciones hacen que IAX sea un protocolo indicado para trabajar bajo redes seguras.

IAX ha sido optimizado para el transporte de voz, lo que ha ocasionado que reciba ciertas críticas por no disponer de un buen sistema de soporte de video. Sin embargo, el carácter abierto del protocolo hace que, en un futuro, pueda mejorarse dicho soporte si los usuarios así lo requirieran.

Como ya se ha comentado, el buen comportamiento que tiene frente a NAT (en contraste con SIP o H.323) otorga a IAX una baza frente a estos protocolos.

2.5. Codecs de compresión de datos de voz

La señal de voz es una señal analógica, mientras que las redes de datos son redes digitales. Por ello, el primer paso en la tecnología VoIP consiste en digitalizar la señal de voz analógica.

El proceso de conversión de la señal de voz analógica en una señal digital (y su proceso inverso) se realiza con un codec (codificador-decodificador). Hay muchas maneras de digitalizar una señal de voz analógica, todas ellas gobernadas por varios estándares. El proceso de la conversión es complejo. Nos limitaremos a comentar que la mayoría de las conversiones se basan en la Modulación Codificada mediante Pulsos (PCM) o variaciones.

Además de realizar el proceso de conversión analógico-digital, el codec se encarga también de la compresión de la secuencia de datos, permitiendo un ahorro del ancho de banda. Este aspecto es especialmente interesante en los enlaces de poca capacidad y permite tener un mayor número de conversaciones VoIP simultáneamente.

2.5.1. G711: aLaw y uLaw

G.711 es el codec básico a partir del cual se derivan casi todos los demás. Es un estándar de la ITU (Unión Internacional de Telecomunicaciones) y fue liberado para su uso público en 1972.

G.711 emplea la técnica PCM (*Pulse Code Modulation*), con una tasa de muestreo de 8 KHz y 8 bits por muestra, operando a 64 Kbps.

Existen dos algoritmos definidos en el estándar, μ-law (utilizado en Estados Unidos y Japón) y A-law (utilizado en Europa). Ambos son logarítmicos, aunque el A-law, que es posterior, fue específicamente diseñado para que fuera más simple de procesar por una computadora ya que tiene mayor rango.

2.5.2. g.729

Este codec, desarrollado por la ITU, tiene muy pocos requerimientos de ancho de banda. El codec estándar trabaja a 8 Kbits/s utilizando un algoritmo conocido como *Conjugate Structure Algebraic Code Excited Linear Prediction* (CS-ACELP).

Además del codec original, existen varias versiones del codec G.729 que es interesante explicar por su extendido uso. La primera de ellas es G.729A. Es una simplificación de G.729. Es menos complejo pero con algo menos de calidad. La segunda versión es G.729B. Consiste en G.729 pero con supresión de silencios, si bien no es compatible con G.729 original ni con G.729A.

Este conjunto de codecs se encuentran protegidos por patentes.

2.5.3. GSM

GSM *Full Rate* es un codec estandarizado por el ETSI (*European Telecommunications Standards Institute*) en 1990 y fue el primer codec de digitalización y compresión de voz utilizado en el sistema de telefonía móvil digital GSM.

GSM *Full Rate* está basado en el algoritmo RPE-LTP (*Regular Pulse Excitation – Long Term Prediction*) y ofrece una salida binaria de 13 Kbps.

La calidad de este codec es inferior a otros estándares existentes hoy en día. Sin embargo, sí fue una muy buena opción en el momento de su estandarización, debido a su buen compromiso entre calidad y complejidad computacional. Actualmente, GSM *Full Rate* se ha sustituido por GSM *Adaptative Rate* o por GSM *Enhanced Rate*, que proveen mayor calidad con menor tasa binaria.

2.6. Tecnologías para implementar una aplicación Web: ECLIPSE y MAVEN

Eclipse es un entorno de desarrollo integrado (*IDE*) de código abierto multiplataforma para desarrollar las llamadas "aplicaciones de cliente enriquecido" (en contraposición a las "aplicaciones de cliente liviano" basadas en navegadores). El IDE utilizado en este proyecto es el de Java, llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse.

Este entorno ayuda a organizar el código y facilita las tareas de compilación, detección y corrección de errores de programación gracias a la automatización de tareas, así como otras características útiles como el auto completado de texto.

Eclipse es un proyecto iniciado por IBM en 2001. En la actualidad es un proyecto desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicio. En la actualidad muchas otras empresas desarrolladoras de software están involucradas en el proyecto Eclipse, como por ejemplo Novell, de la cual hemos tomado las librerías necesarias para la utilización de su API entre Java y OpenLDAP.

Eclipse tiene una comunidad de usuarios que extiende constantemente las áreas de aplicación cubiertas.

En el caso de la IRI, su código fuente se ha desarrollado mediante Eclipse, y su ciclo de vida se gestiona mediante otra herramienta de código abierto denominada Maven.

Entre otras cosas Maven permite administrar varias etapas del ciclo de vida de un proyecto. Por ejemplo, con Maven podemos compilar los componentes de un proyecto, desplegarlos, ejecutar casos de prueba, ejecutar un *clean* (eliminar los directorios donde tenemos desplegado un proyecto para desplegarlo desde cero nuevamente).

Maven asume que contamos con una estructura convenida, en la que espera encontrar los archivos Java del proyecto, por ejemplo:

${basedir}/src/main/java
${basedir}/src/main/resources
${basedir}/src/main/webapp

Maven contiene un módulo de dependencias y un repositorio donde se almacenan tanto los componentes del proyecto como otros componentes de terceros (por ejemplo los archivos *jar* de Apache o las librerías JLDAP de Novell). El repositorio puede residir en un archivo local o bien utilizar directorios compartidos en una Intranet de una compañía. Además Maven cuenta con un repositorio central compartido en Internet.

La información básica de cada proyecto de Maven es gestionada mediante un archivo *xml* denominado *Project Object Manager* (POM); el pom.xml está alojado en la raíz del directorio del proyecto.

En resumen, Maven es una herramienta altamente escalable y configurable, que puede ser integrada con los entornos de desarrollo más populares del mercado.

2.7. Librerías de cliente LDAP para Java: JLDAP

Actualmente existen diversas posibilidades para desarrollar código Java que interactúe con librerías LDAP, mediante librerías y APIs actualmente implementadas. Estas herramientas permiten acceder a directorios LDAP mediante código Java.

Para la realización del proyecto se exploraron varias soluciones: JLDAP, JNDI, Netscape LDAP SDK para Java y Spring. De entre ellas, la que se ha optado por utilizar finalmente en este proyecto es la primera, JLDAP, por ser de fácil implementación y utilización, y por ser además la librería recomendada por OpenLDAP.

JNDI, *Java Naming Directory Interface*, es el estándar de interacción entre Java y los Directorios de datos desarrollado por SDN (SUN Developer Network) y hoy en día perteneciente a Oracle y se dedicó cierto tiempo y esfuerzos a explorar esta alternativa, pero la información disponible en los sitios de Internet consultados para dar soporte a la integración no resultó suficiente para obtener resultados satisfactorios en el tiempo de desarrollo previsto.

Netscape LDAP SDK para Java era la opción sugerida en la bibliografía consultada "*Implementing LDAP*", de Mark Wilcox, pero a parte de en esta referencia no se encontraron mayores referencias a un uso extensivo o generalizado de esta herramienta en otras referencias en Internet, por lo que no se profundizó en esta opción.

Por otro lado, Spring no es únicamente una librería de funciones para integrar LDAP y Java, sino que es toda una plataforma de desarrollo de código java que supone una alternativa o una extensión al modelo EJB. Es decir, no es algo que podamos utilizar añadido a nuestra aplicación desarrollada en Eclipse sino que en principio es una plataforma distinta en la que iniciar el desarrollo de una aplicación similar en Java, que dispondría de herramientas de integración del código creado con librerías LDAP. Esto implica que no nos es útil para el caso que nos ocupa, en el que la aplicación principal ya está desarrollada utilizando otra herramienta.

Tras indicar brevemente los motivos por los que se eligió JLDAP queda señalar que esta elección no significa que una implementación con las otras opciones no sea posible o recomendable. Simplemente desde el punto de vista de la limitación de tiempo a la hora de explorar en detalle cada una de las opciones para la realización de este proyecto, la elección que optimizaba la eficacia en la realización de este proyecto resultó JLDAP.

2.7.1. JLDAP

JLDAP son las siglas de Java LDAP, y son librerías de clases de Java para LDAP desarrolladas por Novell. El último código fuente desarrollado está disponible en el Repositorio CSV de OpenLDAP.

Estas librerías nos permiten escribir aplicaciones para acceder, gestionar, actualizar y buscar información guardada en directorios que sean accesibles mediante LDAPv3 de una forma práctica y sencilla.

La API de JLDAP define interfaces tanto asíncronos como síncronos.

La clase principal de LDAP es la *LDAPConnection*. Esta clase proporciona métodos para establecer conexiones autenticadas o anónimas contra un servidor LDAP, así como los métodos para buscar, modificar comparar y borrar entradas en el directorio.

La clase *LDAPConnection* también proporciona campos para guardar la configuración específica de una sesión LDAP, como por ejemplo el límite en el número de resultados devueltos en una búsqueda o el parámetro de *time out*. Un objeto *LDAPConnection* puede ser clonado, permitiendo que varios objetos compartan una misma conexión de red pero utilizando diferentes configuraciones (mediante el uso de *LDAPConstraints* o *LDAPSearchConstraints*).

Una búsqueda síncrona realizada por un objeto *LDAPConnection* devuelve resultados en un objeto *LDAPSearchResults*, que puede ser enumerado para acceder a las entradas de directorio encontradas. Cada entrada, representada por un objeto *LDAPEntry*, provee acceso a sus atributos devueltos junto a la entrada, representados por objetos *LDAPAttribute*, que pueden ser reproducidos como *arrays* de *bytes* o como *strings*.

El protocolo LDAP provee métodos de acceso al directorio tanto asíncronos como síncronos. Todos los métodos asíncronos toman como entrada y devuelven como salida objetos *Listener*: o *LDAPResponseListener* o *LDAPSearchListener*. Un *Listener* es una cola de mensajes asociada a una petición, y es responsabilidad del cliente leer los mensajes de la cola y procesarlos.

Los mensajes extraídos de un *LDAPResponseListener* son objetos derivados de *LDAPResponse*. Los mensajes extraídos de un *LDAPSearchListener* son objetos derivados de *LDAPResponse*, o bien resultados de búsquedas, o referencias a resultados de búsquedas.

Una búsqueda asíncrona realizada por un objeto *LDAPConnection* devuelve resultados mediante la operación *search* el método *getResponse* del *LDAPSearchListener*.

El método *getResponse* devuelve típicamente un objeto *LDAPSearchResult* que tiene un método *getEntry*, el cual devuelve la *LDAPEntry* que representa la entrada de directorio buscada.

Se supone que ninguna de las clases auxiliares asíncronas va a ser instanciada por un cliente, por lo cual todas ellas carecen de constructores públicos.

Uso de la API de LDAP

Las aplicaciones normalmente utilizan el API de LDAP en cuatro pasos:

1. Construcción de una *LDAPConnection*. Inicialización de una sesión LDAP con un directorio servidor. La llamada *LDAPConnection.connect()* establece un identificador para la sesión, permitiendo que se abran múltiples sesiones simultáneamente sobre diferentes instancias de *LDAPConnection*.
2. Autenticación de un servidor LDAP con un *LDAPConnection.bind()*.
3. Realización de algunas operaciones LDAP y obtención de los resultados. La versión síncrona de *LDAPConnection.search()* devuelve un LDAPSearchResults que se puede enumerar para acceder a todas las entradas encontradas. La versión asíncrona de *LDAPConnection.search()* devuelve un *LDAPSearchListener*, que se utiliza para leer los resultados de la búsqueda. *LDAPConnection.read()* devuelve una única entrada.
4. Cierre de la conexión mediante *LDAPConnection.disconnect().*

El API contiene versiones sincrónica y asíncrona de las operaciones de protocolo LDAP. Los métodos síncronos no terminan hasta que la operación ha finalizado.

Los métodos asíncronos tomar un parámetro de escucha (ya sea *LDAPResponseListener* o *LDAPSearchListener*) y devuelven un objeto *listener* que se utiliza para enumerar las respuestas del servidor. Normalmente se utiliza un bucle para leer el objeto *listener*, que se bloquea hasta que haya una respuesta disponible, hasta que la operación se ha completado.

Se puede compartir un *LDAPResponseListener* entre las operaciones, para multiplexar los resultados. En este caso, el objeto devuelto en una sola operación se pasa a una o varias operaciones.

Para los métodos asíncronos, las excepciones se producen sólo por errores de conexión. Los mensajes resultados de LDAP se convierten en objetos *LDAPResponse* cuyos errores y referencias deben ser controlados por el cliente, mientras que los métodos síncronos lanzan una *LDAPException* en los códigos de resultado distinto de 0.

Para facilitar la realimentación a los usuarios durante las búsquedas síncronas, se pueden obtener resultados intermedios de la búsqueda antes de que la operación de búsqueda completa se finalice, especificando el número de entradas a devolver de cada vez. Se utilizan las enumeraciones estándar de Java para analizar los resultados de las búsquedas síncronas.

Cuando haya un error, se lanzará una *LDAPException* con un código de error específico y el contexto de la información textual específica disponible.

Si se pasa NULL como el valor de un parámetro *LDAPConstraints* o *LDAPSearchConstraints* a una operación, se utilizarán los parámetros por defecto para la operación.

Si se pasa NULL como el valor de un DN a una operación se trata como si fuera una cadena vacía.

La API no distingue entre referencias de continuación de búsqueda LDAP y referencias de LDAP, presentando al cliente una interfaz unificada para el manejo de las dos.

Las implementaciones de la API deben asegurar que la clase *LDAPConnection* es segura para subprocesos.

2.8. Análisis de productos similares en el mercado

Hoy en día existen múltiples soluciones de telefonía IP basadas en Asterisk. El software libre facilita que multitud de empresas de diverso tamaño y origen se dediquen al desarrollo de soluciones adaptadas a las necesidades del mercado.

Las diversas posibilidades de implementación de autenticación de usuarios son las siguientes:

- Utilizar los ficheros básicos de configuración de Asterisk
- Utilizar una base de datos local en la que están almacenados los datos de usuarios y a la que se consulta cada vez que se requiere autenticación.
- Utilizar autenticación contra usuarios creados en un directorio

En 2007 cuando la voz sobre IP comenzaba a popularizarse, multitud de pequeñas y medianas empresas empezaron a desarrollar y ofrecer productos y soluciones de telefonía IP en el mercado. Hoy en día los principales operadores de telefonía así como importantes empresas distribuidoras de *hardware* y servicios de comunicaciones ofrecen soluciones de telefonía IP, principalmente al mercado empresarial.

3. DESCRIPCIÓN E IMPLEMENTACIÓN DEL SISTEMA

El sistema a instalar es complejo, ya que consta de múltiples partes que deben ser instaladas y configuradas de modo que se vayan integrando entre ellas para formar el sistema de telefonía final.

El siguiente esquema muestra de forma simple las tecnologías utilizadas:

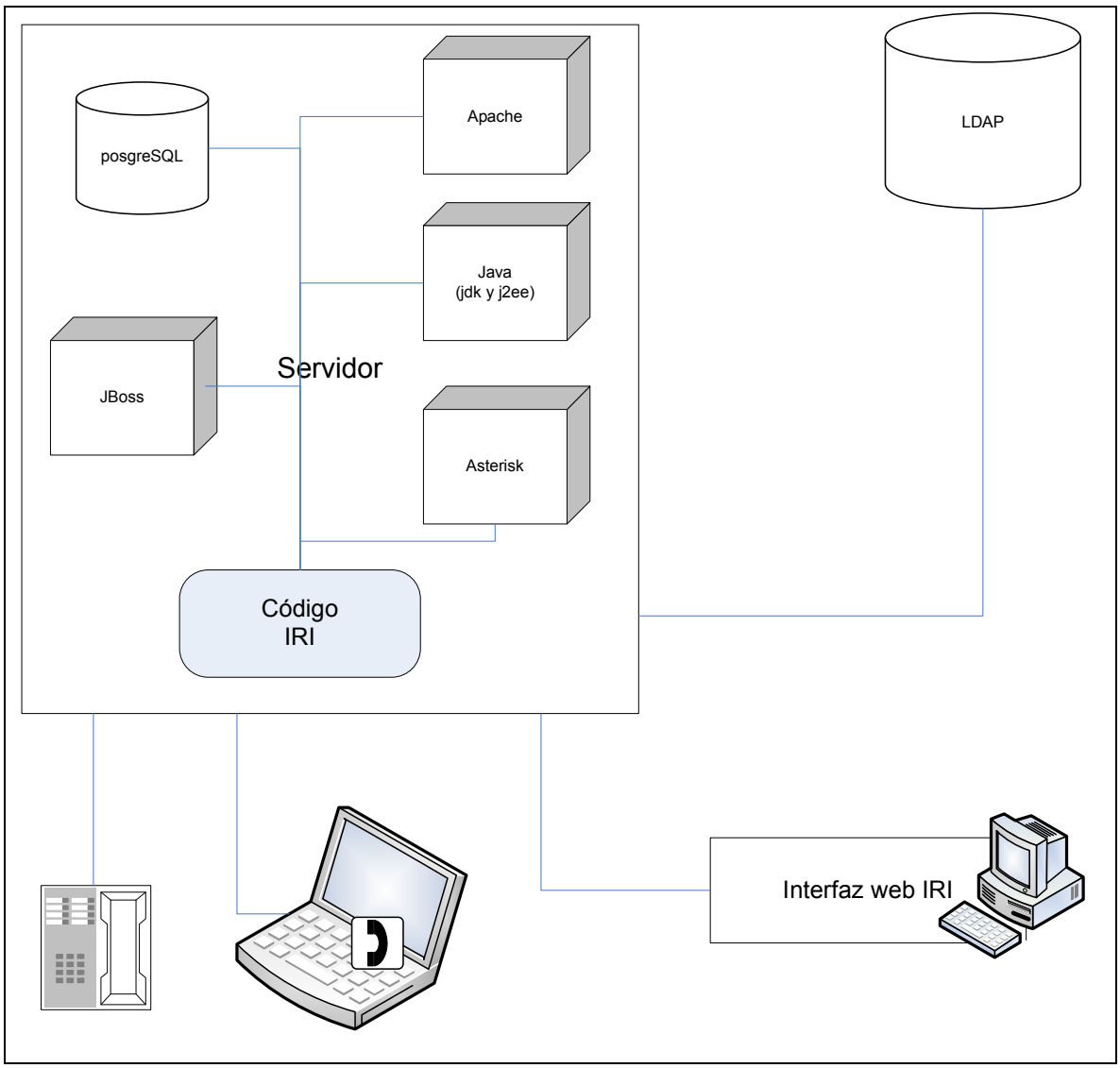


Figura 5. Esquema de integración del sistema

3.1. Requisitos de la aplicación a desarrollar

El requerimiento inicial es que Asterisk se autentique contra LDAP, por lo tanto la decisión inicial es si considerar que los usuarios de telefonía serán nuevos usuarios a crear en LDAP o si podemos reutilizar usuarios existentes a los cuales se añadan los atributos necesarios para Asterisk.

Dado que en el caso de este proyecto no se contaba con un directorio inicial real que sea necesario integrar con Asterisk para hacer las pruebas, se creó un directorio nuevo y se crearon los usuarios necesarios en él. En todo el proceso se han tomado decisiones de diseño intentando que aporten flexibilidad y funcionalidad a la solución final. Las decisiones de diseño iniciales son las siguientes:

- Se supone una base de datos ya existente, por lo tanto no se instalará una nueva base de datos LDAP al instalar las restantes partes del producto. En una ventana de configuración del interfaz Web, se indicarán los parámetros de conexión a dicho directorio. Si la base de datos inicial no existiera, se podría instalar una nueva e indicar sus parámetros en la página de configuración.
- El objetivo del sistema de telefonía es crear usuarios de telefonía Asterisk. Dichos usuarios pueden existir previamente en el directorio o no existir. Si ya existen, se les añadirán los atributos necesarios para dotarlos de telefonía, y si no existen se crearán asumiendo que son personas (ver el siguiente punto) y dotándoles de los atributos necesarios para telefonía.
- Se asumirá por tanto que los usuarios que ya existan en el directorio LDAP están creados en el directorio como personas, y por lo tanto pertenecen a la clase *Person* descrita en el esquema básico de LDAP. Al pertenecer a la clase *person* o a alguna de sus clases hijas (*inetOrgPerson*), poseerán los atributos obligatorios CN y SN
- Al crear un usuario de telefonía, se incluirá a este usuario en una unidad organizativa que permita identificarlo como usuario de Asterisk, es decir, se creará una ou=*usuariosIRI* que será añadida a los usuarios de telefonía IP y permitirá realizar búsquedas sobre ellos con un sencillo filtro.
- Como los nuevos usuarios de telefonía pueden ser usuarios ya existentes en LDAP o ser usuarios completamente nuevos, el proceso de creación de usuarios se basará en una búsqueda filtrada por nombre y apellido (CN) sobre las entradas del directorio. Si el usuario existe, se le incluye en la clase *usuariosIRI* y se le añaden los atributos específicos de Asterisk a sus atributos ya existentes, utilizando la opción *Modify* De LDAP. *Modify* precisa del DN de la entrada a modificar, que se obtendrá mediante la operación *LDAPsearch* anterior. Si el usuario es completamente nuevo, se añadirá una entrada en el directorio que implementará las clases *inetOrgPerson* y *usuariosIRI*, y se definirá el DN de la entrada con un UID igual al “Nombre de Usuario” definido en el formulario Web de recogida de datos de la aplicación.
- La IRI precisa de un usuario administrador inicial que viene preconfigurado. De modo similar, se creará un usuario en LDAP y se le dotará de los permisos de

usuario administrador mediante el uso de listas de acceso de LDAP (ACLs) lo cual se hace directamente sobre LDAP y no mediante la IRI.

  - Se entiende que el usuario que nos proporciona será:
  - O bien el usuario administrador de LDAP si éste va a ser la misma persona que el administrador de Asterisk
  - O bien otro usuario distinto al administrador general de LDAP, pero con los permisos necesarios para buscar a los usuarios de LDAP que serán también usuarios de Asterisk y tener acceso a sus contraseñas

- CN se forma a partir del *givenName* y el *Surname* concatenándolos mediante un símbolo "_", es decir, *givenName_Surname*

Ha sido necesario componer el CN de esta forma para poder utilizarlo como nombre de usuario SIP de Asterisk, ya que éste último no puede contener espacios en blanco.

3.2. Características de la aplicación base

La aplicación de partida consiste en un sistema de telefonía IP basada en Asterisk, que proporciona un interfaz Web de usuario mediante la cual se gestiona el sistema de telefonía IP.

Proporciona una amplia variedad de funciones de telefonía IP, entre ellas:

- Gestión de llamadas SIP
- Interfaz entre Internet y la red pública conmutada
- Funciones básicas y avanzadas de telefonía convencional:
    - Desvío de llamada entrante a otra extensión
    - Transferencia de llamada
    - Captura de llamada a una extensión desde otra extensión
    - Activar modo “No molestar”
    - Conferencia a tres
    - Buzón de voz
    - Agenda
    - Mensajería unificada
    - IVR
    - Planes de numeración
    - Etc.

La integración de la autenticación con LDAP no aporta una nueva funcionalidad a la central de telefonía IP, las funciones arriba descritas continúan efectuándose del mismo modo en que fueron programadas, pero aporta una nueva opción de configuración que permite la flexibilidad adicional de poder elegir el modo de autenticación: mediante Base de datos o por medio de LDAP.

3.3. Componentes y descripción de la aplicación

La aplicación existente consta de los siguientes componentes:

Base de datos PostgreSQL: contiene las tablas que guardan la información de las opciones de configuración de la central de telefonía IP, así como los datos de los usuarios del sistema.

Servidor Web Apache: servidor http de código abierto para enviar páginas Web estáticas y dinámicas en la World Wide Web. Los programadores de aplicaciones Web a veces utilizan una versión local de Apache para previsualizar y probar código mientras éste es desarrollado

Java Development Kit y *Java2 Enterprise Edition*: plataforma Java que proporciona la infraestructura para la creación de aplicaciones de servidor.

Jboss: servidor de aplicaciones J2EE de código abierto implementado en Java.

Asterisk: aplicación de software libre que proporciona funcionalidades de una central telefónica.

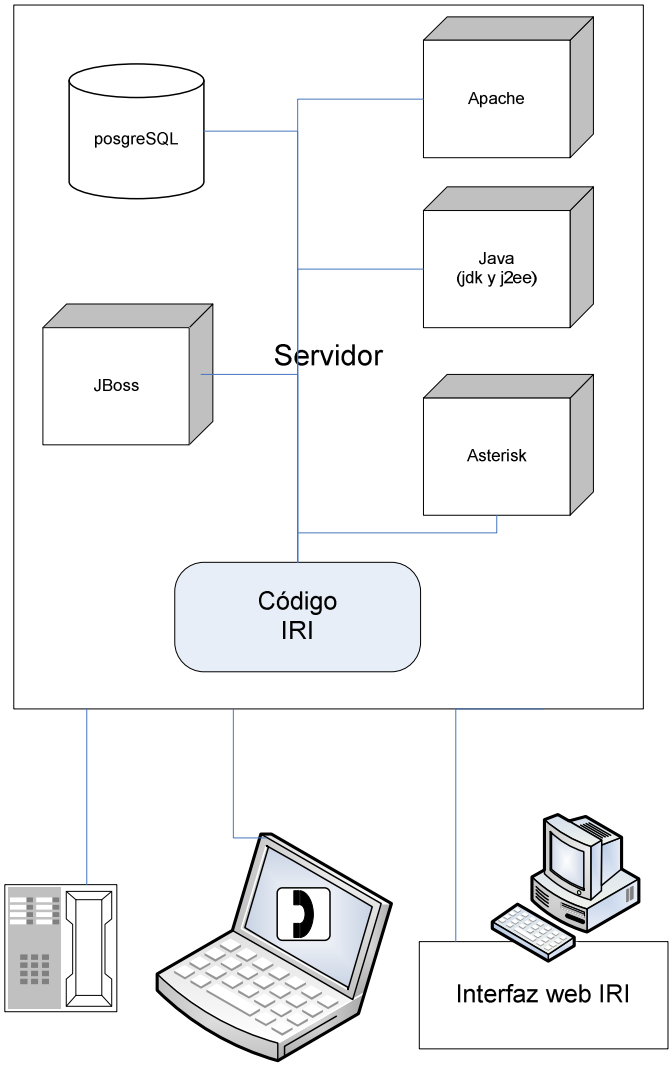


Figura 6. Esquema de integración del sistema

3.4. Estructura de directorios y archivos

Las aplicaciones java J2EE distribuyen en contenedores para gestionar los componentes de la aplicación desarrollada. A continuación se muestra el árbol de directorios de la IRI:

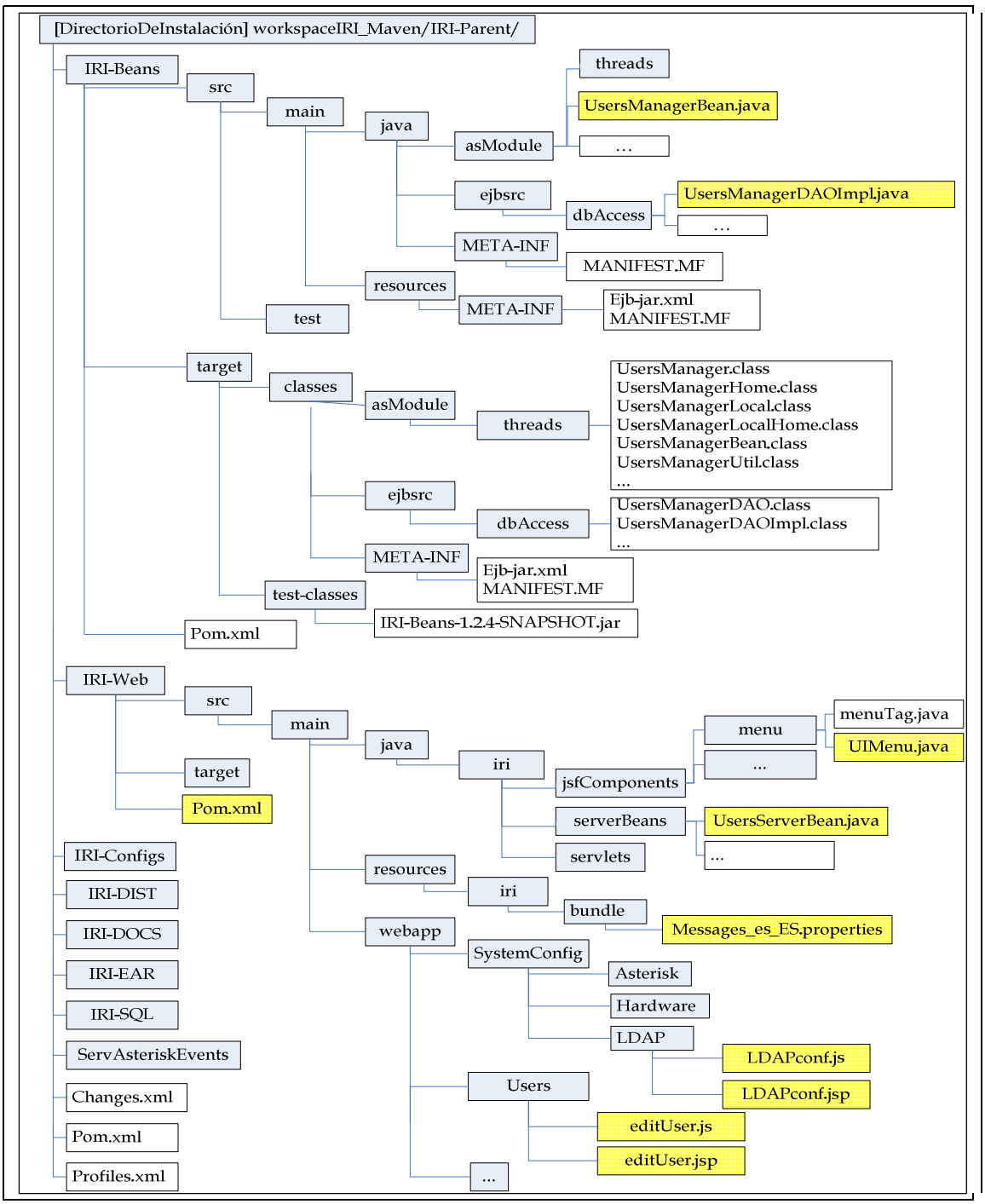


Figura 7. Estructura de directorios y archivos

Un contenedor J2EE se define como "*un periodo de ejecución para gestionar los componentes de la aplicación desarrollados según las especificaciones del API y destinados a proporcionar acceso a los API de J2EE"*.

En el esquema anterior podemos observar los dos contenedores principales de la arquitectura J2EE:

- *IRI-Beans*: un contenedor EJB para albergar componentes Enterprise JavaBean
- *IRI-Web*: un contenedor Web para albergar *servlets* de Java y páginas JSP

Se han destacado en amarillo los ficheros que han requerido modificaciones y desarrollo de código nuevo para integrar la funcionalidad de autenticación contra LDAP.

En el apartado de Implementación se describirán individualmente todos estos ficheros modificados y se explicará la utilidad y finalidad de los cambios realizados.

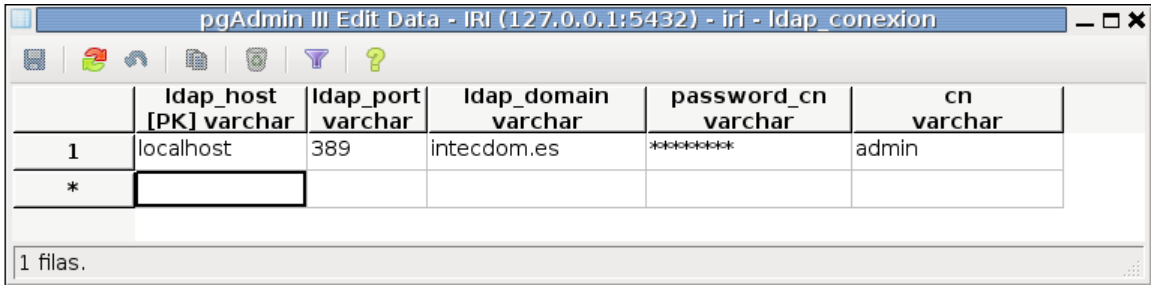
3.5. Tablas de la Base de Datos empleadas

La base de datos utilizada es PostgreSQL. Contiene tablas con la información de las opciones de configuración de la Central de telefonía IP, así como tablas con la información de los usuarios del sistema.

La integración de la central de telefonía IP autenticación mediante LDAP ha supuesto la creación de una vista de configuración para introducir los parámetros de LDAP, así como la creación de una nueva tabla dentro de la base de datos para almacenar estos parámetros.

El hecho de guardar estos parámetros en la base de datos permite mantenerlos en caso de apagado o reinicio del sistema.

Variable	tipo	Descripción
ldap_host	varchar(100)	IP del servidor LDAP
ldap_port	varchar(5)	Puerto de LDAP, por defecto 389
ldap_domain	varchar(100)	Dominio de LDAP a partir del cual buscar o crear usuarios
password_cn	varchar(100)	Contraseña del usuario administrador
CN	varchar(100)	*Common Name* del usuario administrador



3.6. Nuevos métodos desarrollados

El detalle del código correspondiente a los nuevos métodos desarrollados citados a continuación, que constituyen el grueso del trabajo de programación desarrollado, se encuentra en el apartado de anexos incluido en este proyecto. A continuación se enumeran las funciones, así como algunos diagramas de flujo a alto niver para comprender su funcionalidad.

- Ficheros y funciones en el contenedor *IRI-Beans*:
  - **UsersManagerBean.java:**

    Definición de las funciones:
    - getConexionLDAP
    - newConexionLDAP
  - **UsersManagerDAOImpl.java**:

    Implemantación de las funciones:
    - getConexionLDAP

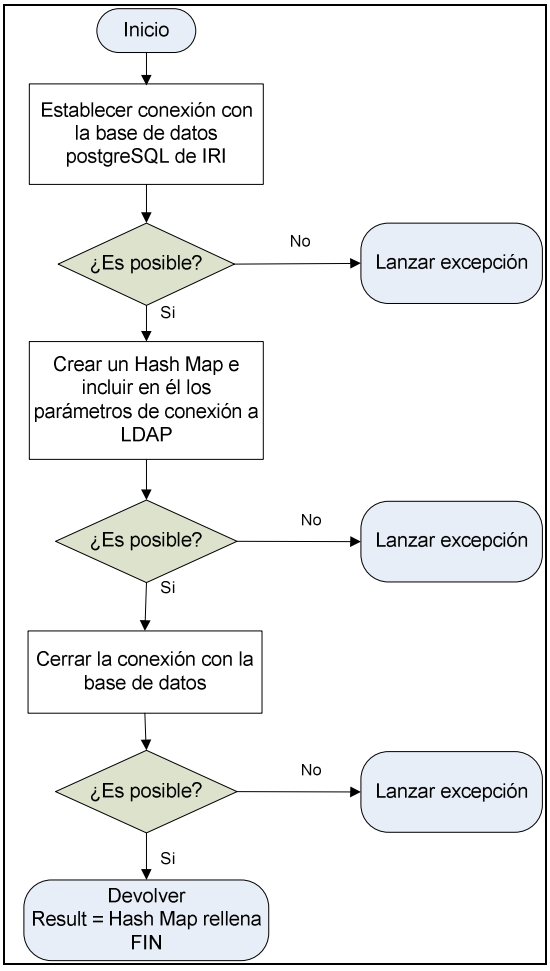


Figura 8. Diagrama de Flujo getConexionLDAP

- newConexionLDAP:

Prepara la llamada para la conexión LDAP añadiendo los parámetros de conexión del *Hash Map*

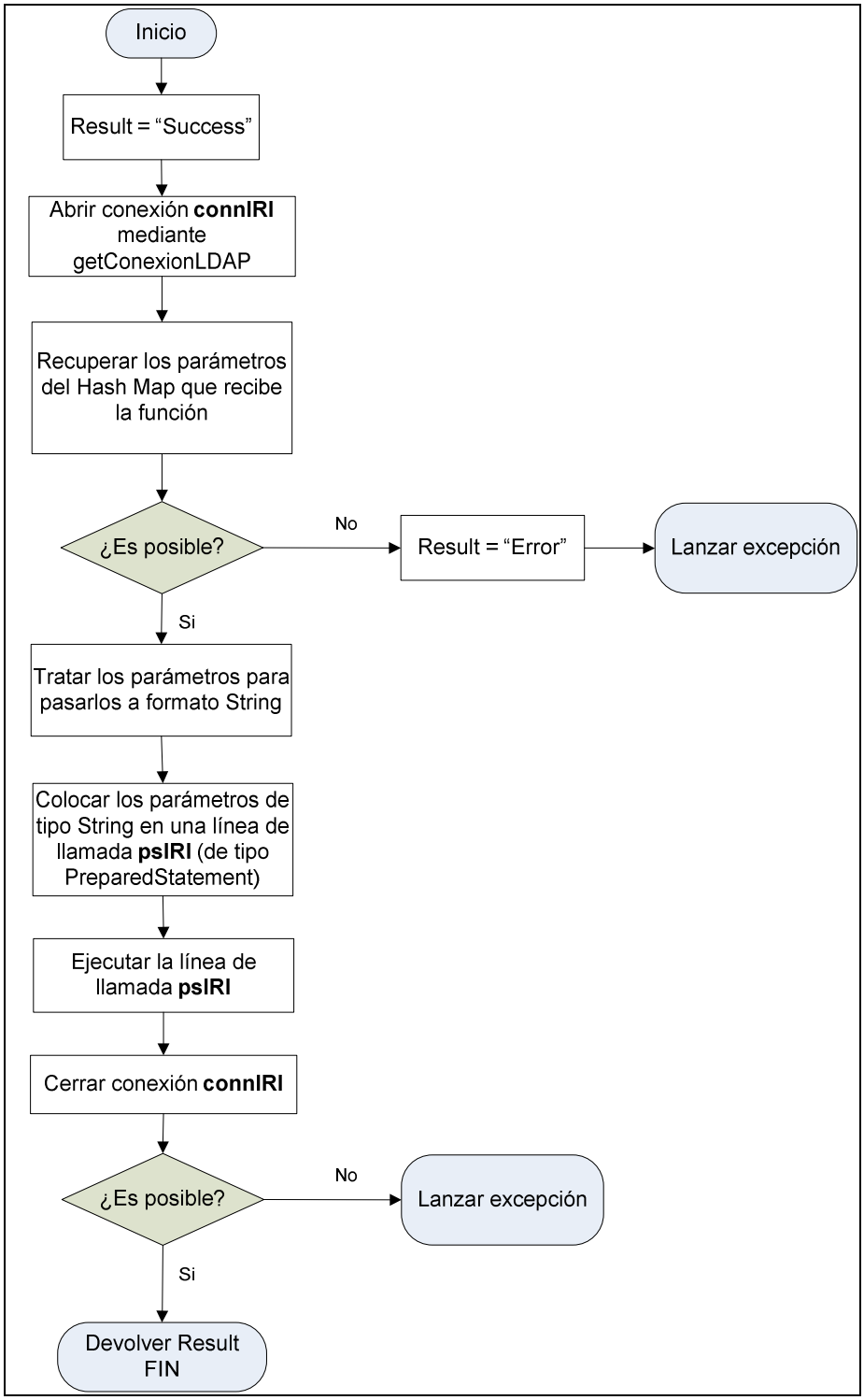


Figura 9. Diagrama de Flujo newConexionLDAP

- Ficheros y funciones en el contenedor *IRI- Web*:
    - **UIMenu.java:**

En este fichero se describe el código correspondiente al menú de la izquierda visible cuando se accede a la interfaz Web mediante el usuario administrador.

Podemos observar los tres submenús: Asterisk, Hardware y LDAP. Estos tres submenús contienen opciones de configuración a las que tan sólo un usuario con responsabilidad de administrador debe tener acceso. Dichas opciones de configuración corresponden a parámetros tales como puerto de escucha de Asterisk, configuración del rango de los puertos RTP, definición de redes IP locales, detección de nuevo *hardware* instalado (Tarjetas DAHDI o ISDN) etc.

En la siguiente imagen podemos ver la nueva pantalla de configuración de LDAP que ha sido añadida para la autenticación por LDAP

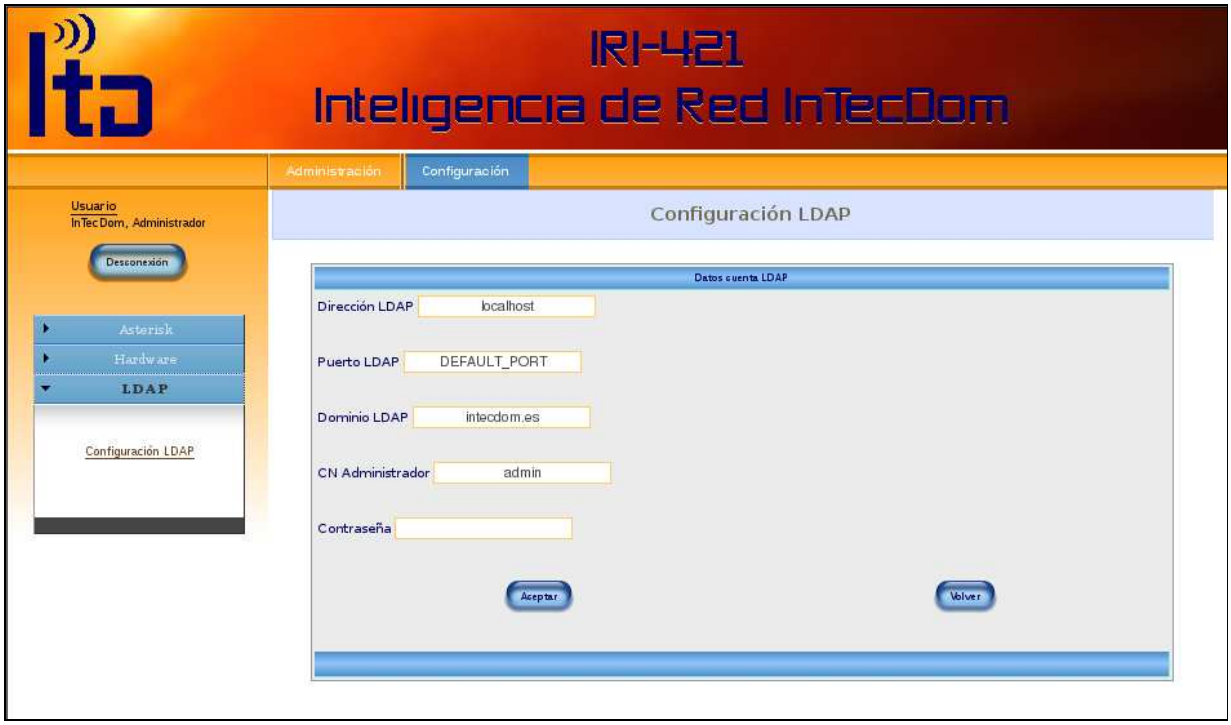


Figura 10. Página Web correspondiente al menú de configuración de IRI

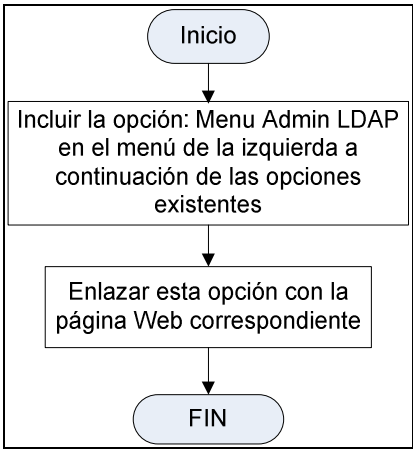


Figura 11. Diagrama de Flujo del código añadido a UIMenu.java

- **UsersServerBean.java:**

Este es el fichero que contiene las principales funciones desarrolladas:

- deleteUserLDAP: dado el DN de una entrada a borrar, conecta al directorio, la busca y la borra.
- newUserLDAP: creamos una nueva entrada en LDAP
- modifyUserLDAP: modificamos una entrada de LDAP

Funciones auxiliares:

- existUserLDAP: comprobamos si existe una entrada en LDAP
- search_dn_UserLDAP_by_cn: a partir del cn de una entrada devuelve el DN de esa entrada
- search_pw_UserLDAP_by_cn: a partir del cn de una entrada devuelve la contraseña de esa entrada
- configLDAP: devuelve una conexión LDAP con los parámetros definidos en la entrada
- getConexionLDAP: devuelve un *Hashmap* con los parámetros de la conexión LDAP

- deleteUserLDAP:

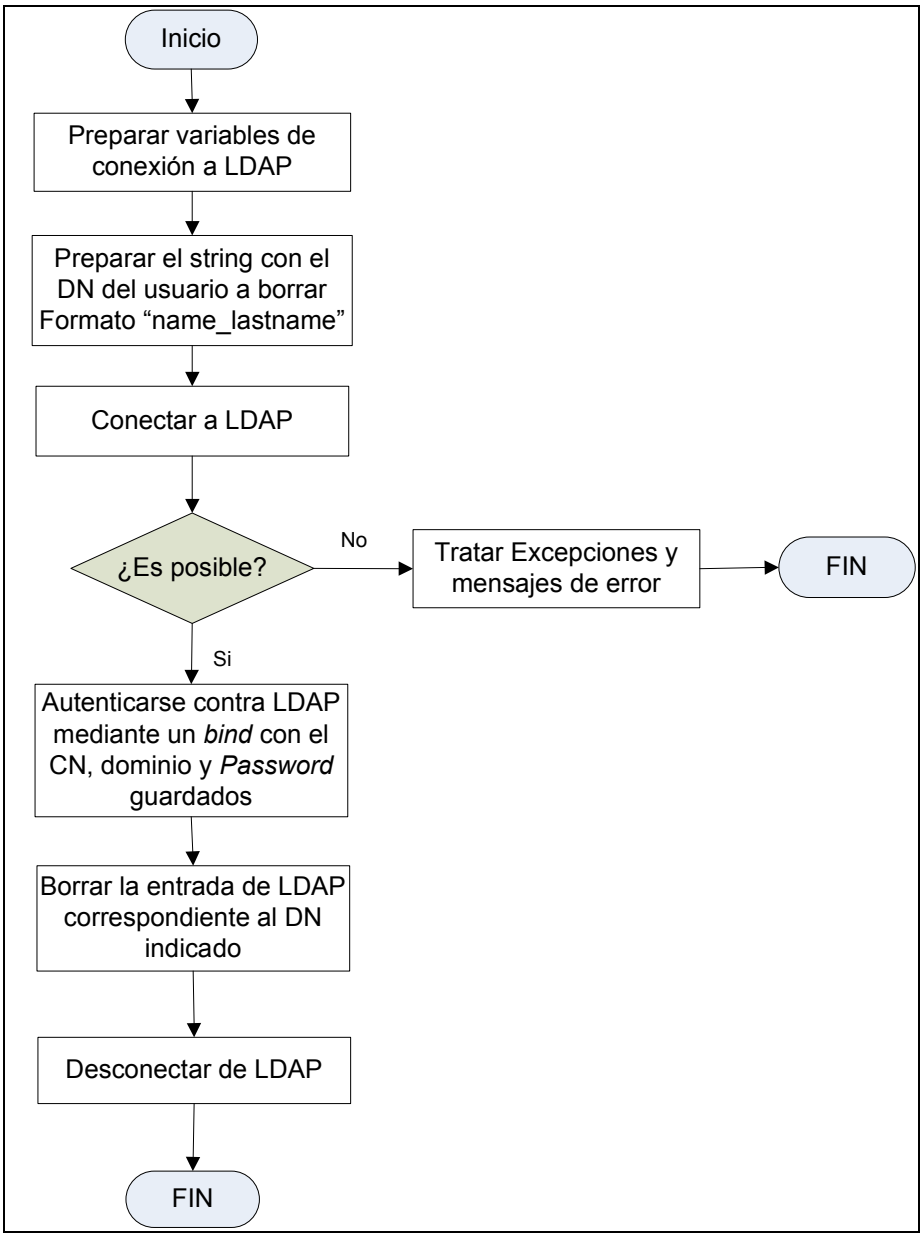


Figura 12. Diagrama de Flujo deleteUserLDAP

- newUserLDAP:

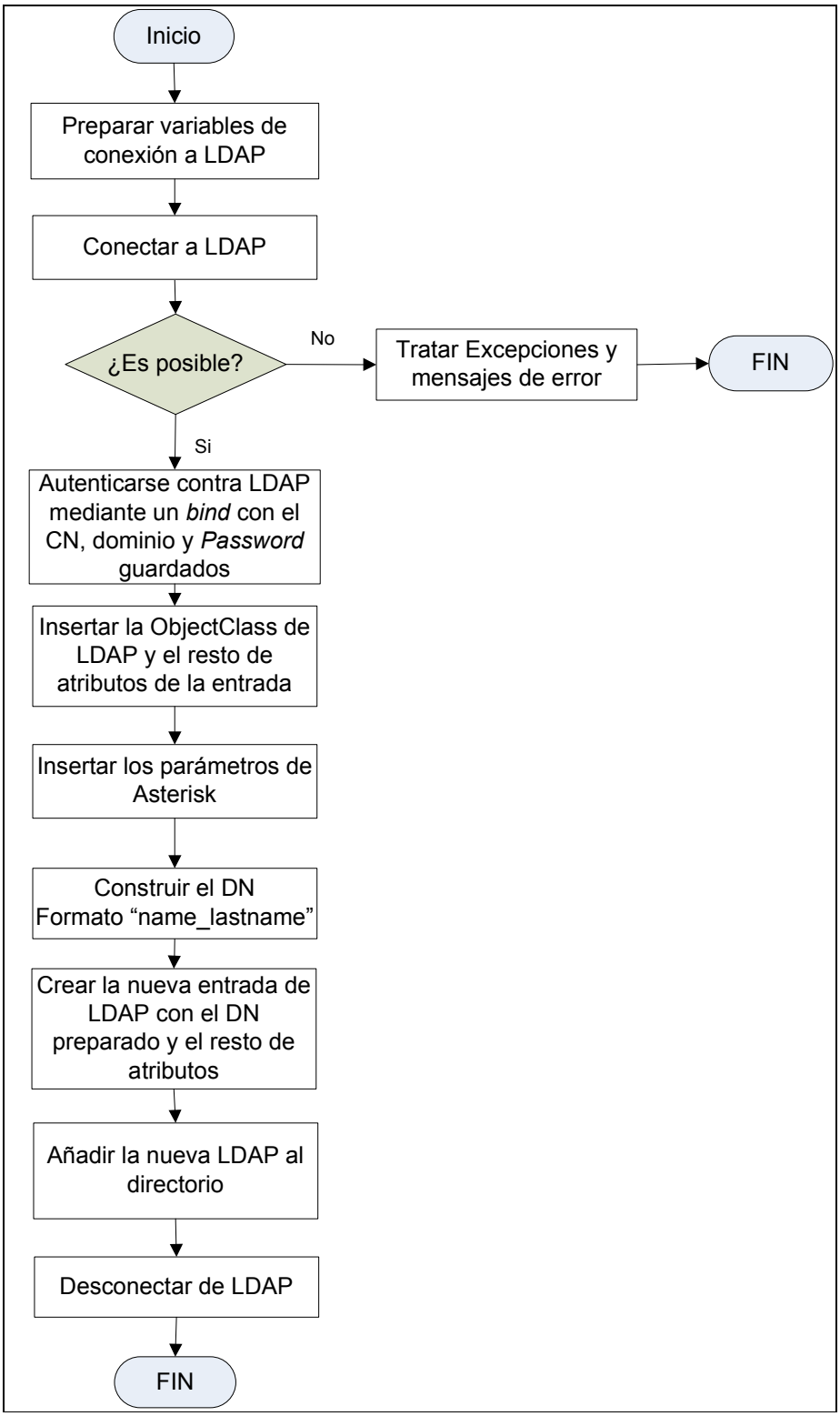


Figura 13. Diagrama de Flujo newUserLDAP

- modifyUserLDAP:

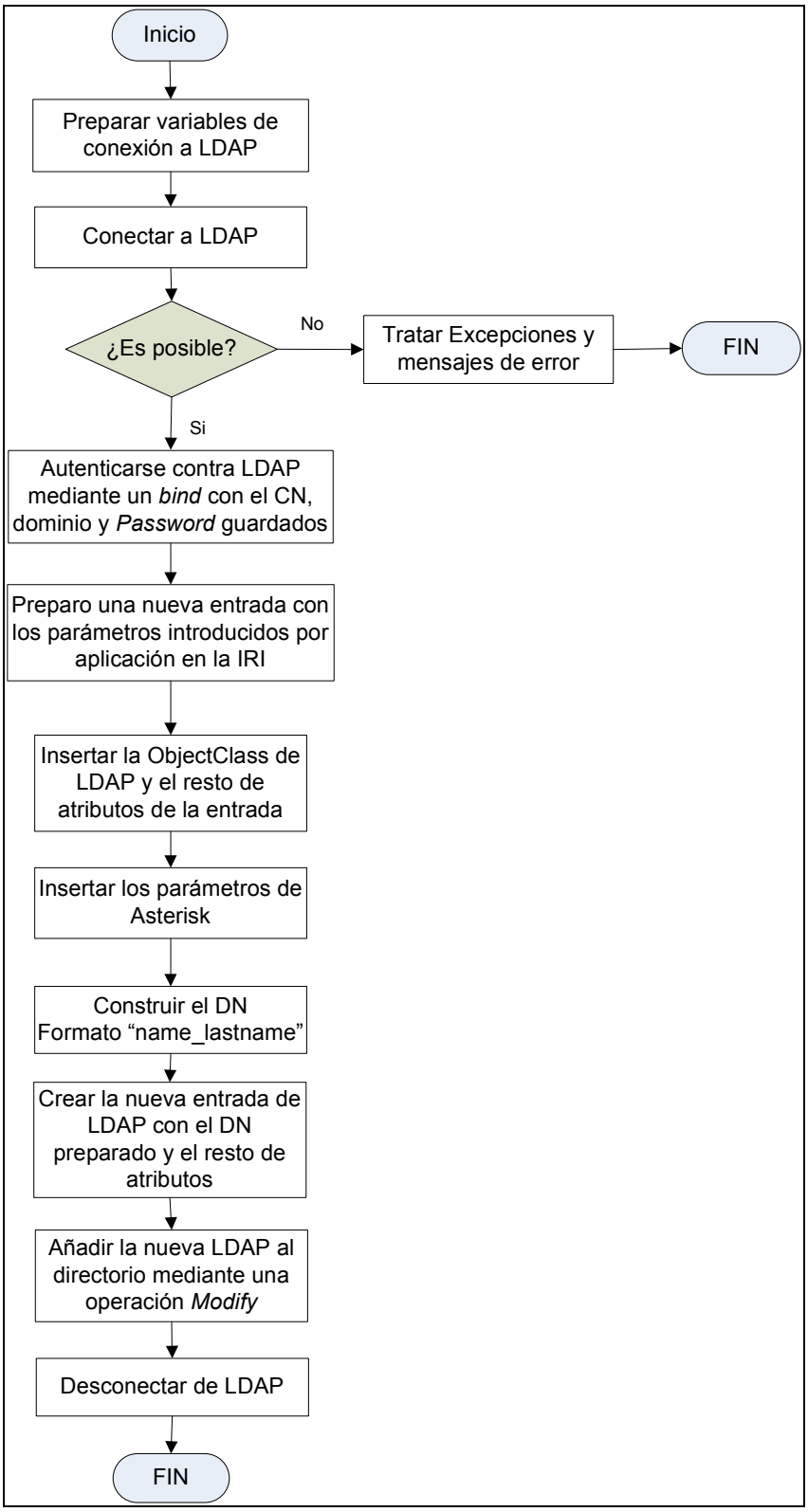


Figura 14. Diagrama de Flujo modifyUserLDAP

- **Messages_es_ES.properties:**

Mensajes lanzados en las excepciones

- **editUsers.js:**

Este código se ejecuta en las vistas de administración de usuarios, por lo tanto en él se utilizan todas las funciones definidas anteriormente para realizar la autenticación contra LDAP en las operaciones de creación, búsqueda, modificación y borrado de usuarios.

Los datos de un usuario se guardan en una variable de tipo ***Hashmap***, con los siguientes campos:

Campo	Tipo	Descripción
UserName	String	identificador de usuario
id_profile	Integer	Perfil de usuario: 1 = administrador (acceso total) 2 = usuario (acceso de usuario)
id_type	Integer	Perfil de usuario: 1 = friend (llamar y recibir) 2 = peer (sólo llamar) 3 = user (sólo recibir)
id_language	Integer	Idioma de usuario: 1 = Español 2 = Inglés
id_group	Integer	Grupo de usuario: 1 = grupo_test
Name	String	Nombre
Lastname	String	Apellidos
Password	String	Contraseña
confirmPassword	String	Contraseña
Email	String	Correo electrónico
Canreinvite	String	Opción de telefonía canreinvite
Nat	String	Opción de telefonía nat
Voicemail	String	Buzón de voz
Codecs	ArrayList de String	Codecs: 1 = alaw 2 = ulaw 3 = g729

En el sistema de telefonía existente, los datos de los nuevos usuarios son rellenados en la página Web y desde ahí son leídos por diversas funciones y guardados en los campos de la tabla *hash* de la variable usuario.

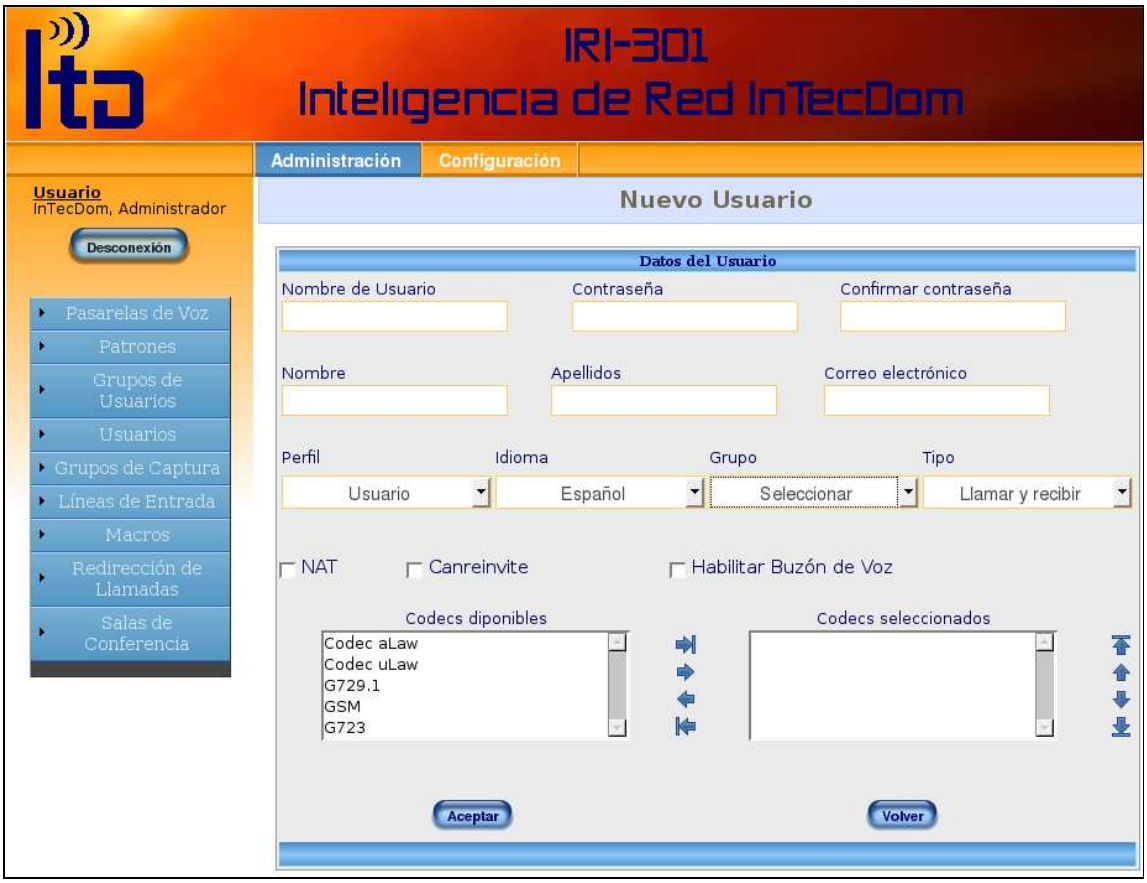
Este mismo funcionamiento se ha aplicado para LDAP. Los datos almacenados en la tabla *hash* de la variable ***usuario*** son utilizados para realizar la conexión y autenticación con el directorio.

Como vemos en la tabla *hash* descrita anteriormente, hay varios campos cuyo valor está limitado a ciertas opciones que están numeradas y almacenadas en distintas tablas de la base de datos PostgreSQL. A la hora de desarrollar el código para LDAP se prefirió guardar estas variables como texto en lugar de como enteros. De este modo la información almacenada en LDAP será más descriptiva.

El fichero **editUsers.js** contiene las funciones:

- *getUserInfo,*
- *getUserCodecs,*
- *isUserCodec,*
- *getProfiles,*
- *getLanguages,*
- *getGroups,*
- *getUserTypes,*
- *getUserCodecs*

Todas ellas permiten recoger la información de usuario que ha sido introducida por medio de la interfaz Web en la página mostrada a continuación y guardada en la tabla *hashmap* anterior



- **LDAPconf.js:**

Contiene las funciones:

- *onLoad*
- *ApplyChanges*

Ambos ficheros **editUsers.js** y **LDAPconf.js** pertenecen respectivamente a *Users* y *SystemConfig*, que a su vez forman parte de los java *servlets* contenidos en el fichero *WebApp* correspondiente al contenedor *IRI-Web*.

- **pom.xml:**

Por último detallamos las modificaciones efectuadas en el fichero pom.xml, que se utiliza para identificar la información de proyecto Maven.

En este fichero la modificación efectuada por el desarrollo de este proyecto consiste en añadir una dependencia a la librería Novell de LDAP, y añadir el “artefacto” Maven jldap de versión 4.3

3.7. Problemas encontrados y soluciones propuestas

Dividiremos esta sección según las fases del proyecto, desde la primera fase de estudio del estado del arte hasta la fase de pruebas, describiendo los problemas encontrados en cada una de las fases y presentando las soluciones y decisiones adoptadas en cada caso.

Las últimas dos fases de Análisis final de trabajo y costes y Documentación y revisión, se realizaron tomando como referencia análisis hechos habitualmente en otros proyectos de telecomunicaciones.

1. Estudio del estado del arte de las tecnologías implicadas

En esta fase hubo que decidir las diferentes tecnologías a utilizar en la integración, así como las versiones de las mismas que se instalarían.

El capítulo 2 de este documento describe en detalle las tecnologías utilizadas y realiza una comparativa de las distintas opciones en el mercado. Tanto LDAP como la solución desarrollada en Java que implementa Asterisk eran elementos de partida.

Cabe señalar como principal problema en esta fase la elección de las librerías de cliente LDAP para java. Tanto Spring como el JDK de Netscape fueron descartados, el primero por no adaptarse a los requisitos necesarios ya que no es sólo un medio de conectar Java con LDAP sino que es todo un marco de desarrollo, es decir, podría sustituir a Eclipse en la parte de desarrollo Web y que ya integraría sus propios métodos de integración con LDAP, pero que no aplicaría en este proyecto ya comenzado en Eclipse. En el segundo caso el JDK de Nestcape fue descartado porque la información encontrada sobre él fue bastante escasa y bastante antigua.

Con JNDI se hizo un intento inicial de implementación, pero se encontraron problemas para serializar y des-serializar correctamente las entradas de LDAP .

En paralelo se comenzó a implementar JLDAP, que además de estar recomendado por OpenLDAP mostró una mayor facilidad de uso que el anterior, por lo que finalmente se optó por esta solución para la integración.

2. Definición de los requisitos de la nueva herramienta

Partiendo del requisito inicial de autenticar a los usuarios de telefonía IP del sistema en LDAP, se presentaban muchas opciones de materializar esa integración sobre todo dado el hecho de no partir de un árbol de datos concreto de directorio. Se comentarán las principales cuestiones.

Definición de los parámetros iniciales de LDAP en la IRI

Para la conexión inicial a LDAP, es necesario que se definan en la IRI los parámetros de conexión, tales como el *host* donde reside el servidor LDAP, el puerto de acceso y las credenciales del usuario administrador de LDAP.

Las posibilidades barajadas fueron la creación de una página inicial en la que se piden estos datos y se realiza la conexión a LDAP, o el acceso inicial a la aplicación mediante el usuario administrador preconfigurado, y una vez dentro del menú de administración añadir una ventana dentro del menú de configuración en la que se introduzcan y se envíen estos parámetros.

Se consideró que la segunda opción es más apropiada dado que mantiene el aspecto de la aplicación existente e integra la nueva opción de configuración en un entorno accesible para su posible revisión en el futuro.

Creación inicial de usuarios frente a volcado inicial

Al principio se pensó en la posibilidad de hacer un volcado de todos los usuarios de LDAP en la IRI, pero esta idea presentaba los siguientes problemas:

- Los usuarios del directorio LDAP existente pueden no tener ningún atributo concreto que determine el tipo de plan de llamada que deberán utilizar.
- En el caso anterior, aunque pudiéramos identificar algún atributo, es requisito para la IRI que los grupos de llamada que implementan los distintos contextos estén creados con anterioridad a la creación de los usuarios, ya que el grupo elegido es un atributo requerido en la IRI para los nuevos usuarios.

Tomando en consideración los problemas anteriores, así como el hecho de que la asignación de extensiones será individual y deberá hacerla el administrador de forma manual en la aplicación, se decidió que la creación de usuarios no se haría mediante volcado inicial, sino individualmente. Es decir, cuando se quiere crear un nuevo usuario en la IRI, se introducen sus credenciales y si el usuario ya existe, se le añaden los nuevos atributos necesarios para la telefonía IP en su misma entrada de LDAP. Si el usuario no existe, se creará nuevo en LDAP, dentro de un grupo de usuarios de Asterisk creado inicialmente.

3. Estudio de la herramienta existente

En este paso los problemas principales fueron el aprendizaje de la herramienta Eclipse para poder desarrollar e incluir las nuevas funcionalidades al desarrollo software ya existente, así como la familiarización con la herramienta Maven2. Cabe comentar que el estudio de ambas herramientas no ha sido exhaustivo, sino que se limita al acercamiento a las mismas necesario para adquirir los conocimientos a nivel usuario que permitan la realización de este proyecto.

4. Desarrollo de la herramienta

En esta fase el principal problema al que enfrentarse es el tiempo. El proyecto comprende un abanico tan amplio de tecnologías en constante desarrollo, que pretender alcanzar un conocimiento profundo sobre todas y cada una de ellas resultaría difícilmente abarcable.

Frente a este problema la solución propuesta consiste en un análisis crítico previo sobre las etapas a desarrollar, y una priorización de contenidos. Es decir, se ha decidido priorizar el estudio en más profundidad primero la tecnología LDAP, que supone la innovación en este proyecto de integración, y en segundo término la tecnología Asterisk, que supone la base de la centralita telefónica existente.

Por lo tanto como se ha comentado anteriormente, el desarrollo de la herramienta se ha llevado a cabo tras un amplio estudio sobre las tecnologías LDAP y Asterisk, y un estudio más reducido sobre el resto de tecnologías necesarias para la integración.

5. Preparación de la maqueta instalable del producto final

Los principales problemas encontrados en esta fase del proyecto fueron logísticos. Una gran parte de la aplicación fue desarrollada en las instalaciones de la empresa Intecdom, en las cuales tuve acceso a la intranet de la empresa, a los repositorios centrales conteniendo el proyecto IRI, del cual se preparó una "rama" separada para integrar este proyecto, así como a los elementos físicos necesarios para la maqueta, por ejemplo un teléfono hardware IP.

Otra parte del desarrollo se realizó remotamente, para ello fue necesario instalar una VPN en el ordenador en el que desarrollaba el software, para de este modo poder tener acceso desde el exterior a la intranet y al repositorio central de la empresa.

6. Batería de Pruebas

Durante las pruebas de la maqueta, que de igual modo se realizaron tanto en Intecdom como en el exterior, continuamos con problemas logísticos. Otro elemento que facilitó la realización de las mismas fue la instalación en un ordenador doméstico con sistema operativo Windows XP de una máquina virtual en la cual se instaló una distribución Debian, para de este modo disponer de un soporte en el que instalar un directorio LDAP, y poder realizar en él tanto pruebas de funcionalidad de LDAP (creación, modificación y borrado de usuarios y grupos, creación y uso de listas de control de acceso, y otras pruebas de configuración) como pruebas de conectividad entre la centralita telefónica y un directorio externo.

Toda la instalación del software de la máquina virtual (*VMware Player 2*) así como del sistema operativo Debian y de OpenLDAP se encuentra detallada en los anexos de este documento.

Otro elemento que facilitó los problemas logísticos fue la utilización de *softphones* en lugar de teléfonos IP hardware. Los *softphones* utilizados fueron X-Lite, un software distribuido por la empresa Counterpath Corporation, de la cual se utilizó una versión libre del software que cuenta con unas características básicas de telefonía IP.

4.PRUEBAS

El cambio propuesto en la configuración de la centralita telefónica existente supone básicamente el cambio de la utilización de los ficheros de configuración de Asterisk a la utilización de LDAP.

La autenticación de usuarios por defecto en Asterisk se realiza mediante el fichero de configuración Sip.conf. Este fichero contiene la información de los usuarios que están dados de alta en la centralita telefónica, y se accede a este fichero para autenticar a los usuarios a la hora de iniciar una llamada telefónica desde la centralita, o bien para realizar tareas administrativas en el fichero.

La introducción del módulo res_ldap en Asterisk y la posterior conexión al servidor LDAP permiten sustituir tanto las tareas administrativas como la verificación inicial de la información de usuario en el fichero sip.conf por una verificación y acceso directamente en LDAP.

Como se acaba de comentar, los principales casos de uso analizados serán por lo tanto el inicio de conexión de las llamadas y las tareas de administración de usuarios de la centralita, tales como creación, modificación y borrado de usuarios.

Las pruebas realizadas corresponden a las siguientes fases y casos de uso:

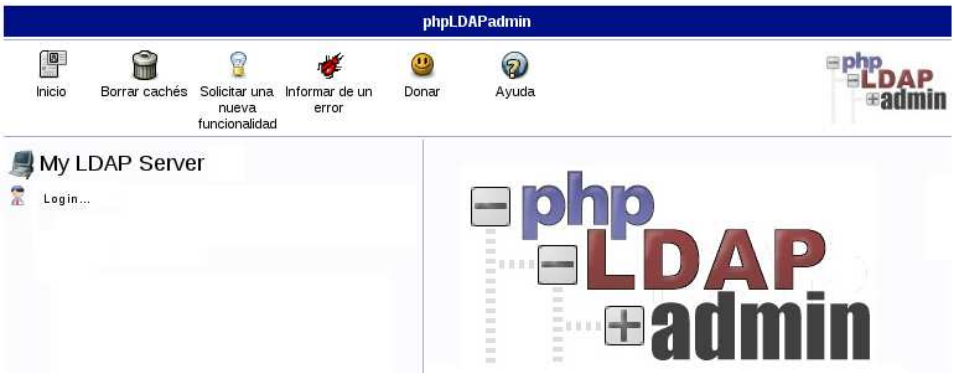
- Fase 1: Inicialización del directorio LDAP en la centralita telefónica
- Fase 2: Actividades cotidianas de la centralita telefónica
    - Administración de los usuarios de la centralita:
        - Creación de un usuario en la centralita telefónica
            - Si el usuario aún no existe en LDAP
            - Si el usuario ya existe en LDAP
        - Modificación de un usuario
        - Borrado de un usuario
    - Autentificación en el inicio de llamadas

A continuación se detalla cada uno de estos casos de uso probados.

4.1. Fase 1: Inicialización del directorio LDAP en la centralita telefónica

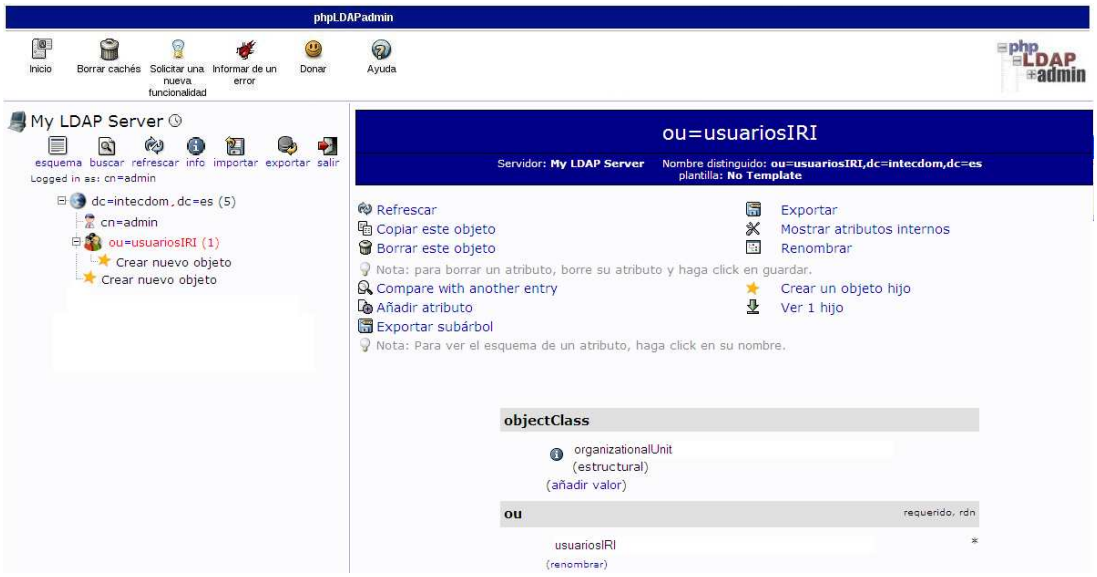
Para el testeo de las funcionalidades de autenticación de la centralita y de LDAP, partimos de un directorio LDAP que nosotros mismos hemos instalado; el proceso de instalación está detallado en los apéndices de este documento.

Para facilitar el acceso directo al nuevo servidor LDAP, utilizaremos la aplicación phpLdapAdmin (PLA). PLA es una herramienta basada en interfaz Web, cuya finalidad es la administración de servidores LDAP. Está escrita en PHP y funciona en varias plataformas. Se encuentra disponible bajo licencia GPL. Posee una vista jerárquica basada en árbol en donde se puede navegar por toda la estructura de directorio. Permite ver los esquemas LDAP, realizar búsquedas, crear, borrar, copiar y editar entradas LDAP, incluso copiar entradas entre servidores LDAP:

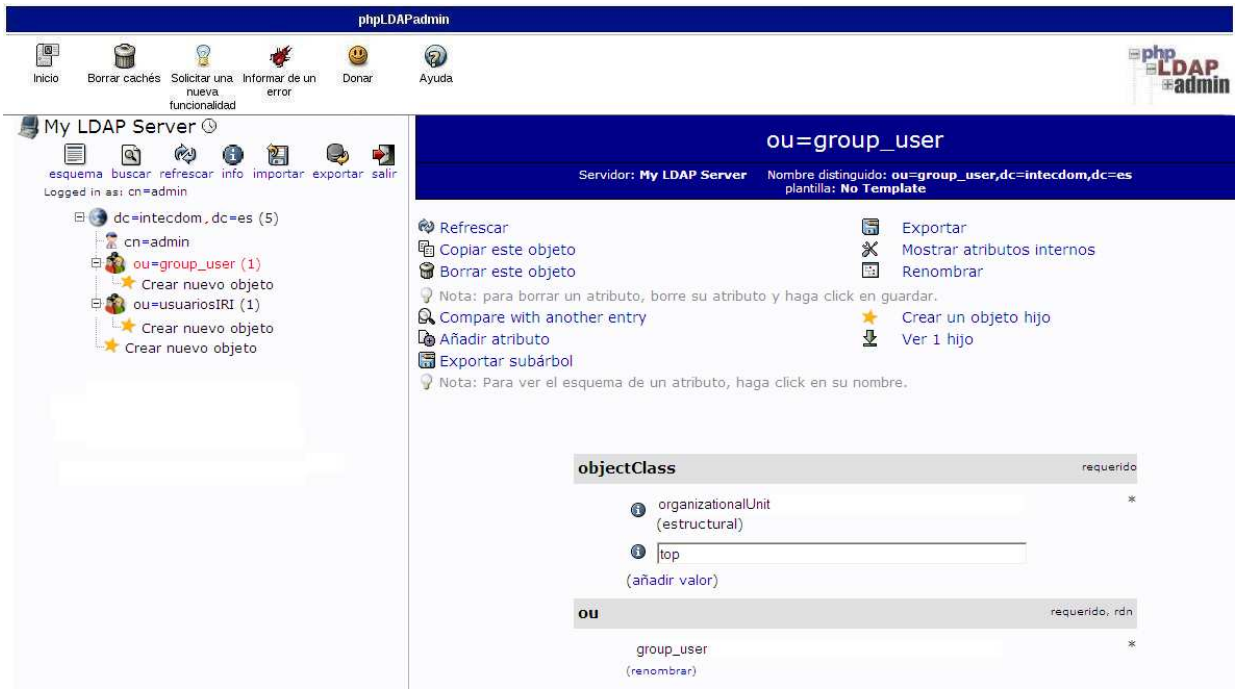


En el nuevo directorio LDAP creamos un árbol de información que contiene ciertos usuarios, para posteriormente reutilizar estos usuarios en nuestras pruebas con la centralita telefónica.

Primeramente creamos un objeto de tipo ObjectClass cuyo atributo “ou” tiene el valor *usuariosIRI*: éste será el grupo al que pertenecerán todos los usuarios que creemos desde la IRI.



A continuación creamos otro grupo distinto, llamado *group_user*, de las mismas características que el anterior:



Por último, creamos un usuario dentro de este nuevo grupo. Este usuario lo crearemos con los atributos *inetOrgPerson*, que hereda los atributos obligatorios de la clase Person: cn y sn.

Una vez disponible la centralita telefónica de partida basada en Asterisk, accedemos a ella con el usuario administrador por defecto que queda preinstalado durante la instalación de la centralita. Los pasos necesarios para la instalación de la centralita están detallados en uno de los anexos de este documento.

Al acceder con un usuario administrador, tenemos directamente acceso a las vistas de administración del sistema. Seleccionamos la opción *Configuración* dentro del menú superior:

Una vez dentro del menú de *Configuración*, seleccionamos en el menú izquierdo la opción de *Configuración de LDAP*:

Las opciones de configuración requeridas para el uso de LDAP son las siguientes:

- **Dirección LDAP:** la dirección IP del servidor en el que está instalado LDAP.

  Si el servidor LDAP fuera de grandes dimensiones y estuviera distribuido en diversos servidores (como se vio en el apartado correspondiente al modelo de nombrado en este documento), podría indicarse la dirección de uno de ellos y posteriormente LDAP distribuiría la solicitud al servidor requerido al estar todos ellos enlazados mediante referencias (*referals*).

  Si el servidor LDAP estuviera instalado en el mismo servidor físico en el que se ha instalado la centralita telefónica, podríamos indicar el valor *localhost* en este campo.

  - **Puerto LDAP:** el puerto por defecto de LDAP es el puerto 389.

  En la realización de esta vista de administración de la centralita telefónica el valor de este puerto está guardado como valor por defecto en la base de datos postgreSQL, en la tabla ldap_conexion descrita en el apartado *IMPLEMENTACIÓN* de este documento.

  Si el servidor LDAP existente tiene configurada la conexión mediante protocolo seguro HTTPS, el puerto por defecto en este caso es el 636 y habría que introducir el valor en este campo.

  Para cualquier otra configuración personalizada de LDAP en el que el puerto de acceso variase respecto al puerto por defecto, habría que introducir el valor del puerto en este campo.

- **Dominio LDAP**: el dominio base sobre el que vamos a realizar las búsquedas de usuarios a autenticar.

  Normalmente será el nombre del dominio de la empresa en la que se instala la centralita telefónica.

  En el caso de que la empresa tenga un subdominio en el que guarda el registro de usuarios, se indicaría directamente este subdominio para mayor eficiencia de las búsquedas en el directorio.

- **CN Administrador**: el atributo *Common Name* del usuario del directorio con privilegios sobre los usuarios que queremos integrar en la centralita telefónica.

  Como ya se ha comentado, las búsquedas de usuarios en directorios están basadas habitualmente en el atributo CN o *Common Name*. Necesitamos incluir en la centralita telefónica las credenciales de un usuario del directorio que tenga acceso a los usuarios que queremos registrar en al centralita, con privilegios suficientes tanto para realizar una operación de *bind* como para crear, modificar o borrar usuarios en el directorio.

  Por seguridad se recomienda crear en el directorio un usuario distinto del administrador general, por ejemplo un usuario admin_VoIP, para poder hacer el seguimiento del usuario responsable de los cambios realizados en los usuarios del directorio si fuera necesario.

- **Contraseña**: la contraseña correspondiente del usuario administrador cuyo CN hemos introducido en el campo anterior.

Una vez introducidos todos los datos de configuración, pulsamos el botón Aceptar y quedará definida la autenticación mediante LDAP, sustituyendo a la opción de autenticación mediante ficheros de configuración de Asterisk definida por defecto en el software de la centralita telefónica.

Si fuera necesario reutilizar la centralita telefónica y conectarla a un directorio LDAP diferente, se puede acceder a esta misma página de configuración y actualizar los valores de los campos necesarios. No obstante, esto borraría los valores anteriores, no siendo posible establecer conexiones a servidores LDAP diferentes al mismo tiempo.

4.2. Fase 2: Actividades cotidianas de la centralita telefónica

Tras la conexión de la centralita telefónica con el directorio LDAP, procedemos a las pruebas de administración y de uso de los usuarios que se deben autenticar contra LDAP.

4.2.1. Administración de los usuarios de la centralita

Como se ha comentado, las operaciones administrativas de usuarios de la centralita posibles son la creación, modificación y borrado. Todas estas operaciones ya estaban soportadas en la centralita telefónica existente basada en la utilización de base de datos postgreSQL mediante el uso de tablas de la base de datos en la que se almacenaban los usuarios y sus opciones de configuración, y a las que se accedía para actualizar cualquiera de los parámetros configurables.

En la configuración de uso de LDAP, se reutilizan las funciones existentes y se modifican para añadir la autenticación previa contra el directorio. Posteriormente, las modificaciones de los usuarios y sus parámetros configurables serán hechas paralelamente tanto en las tablas de postgreSQL como en el directorio LDAP, para mantener la coherencia de datos entre los dos.

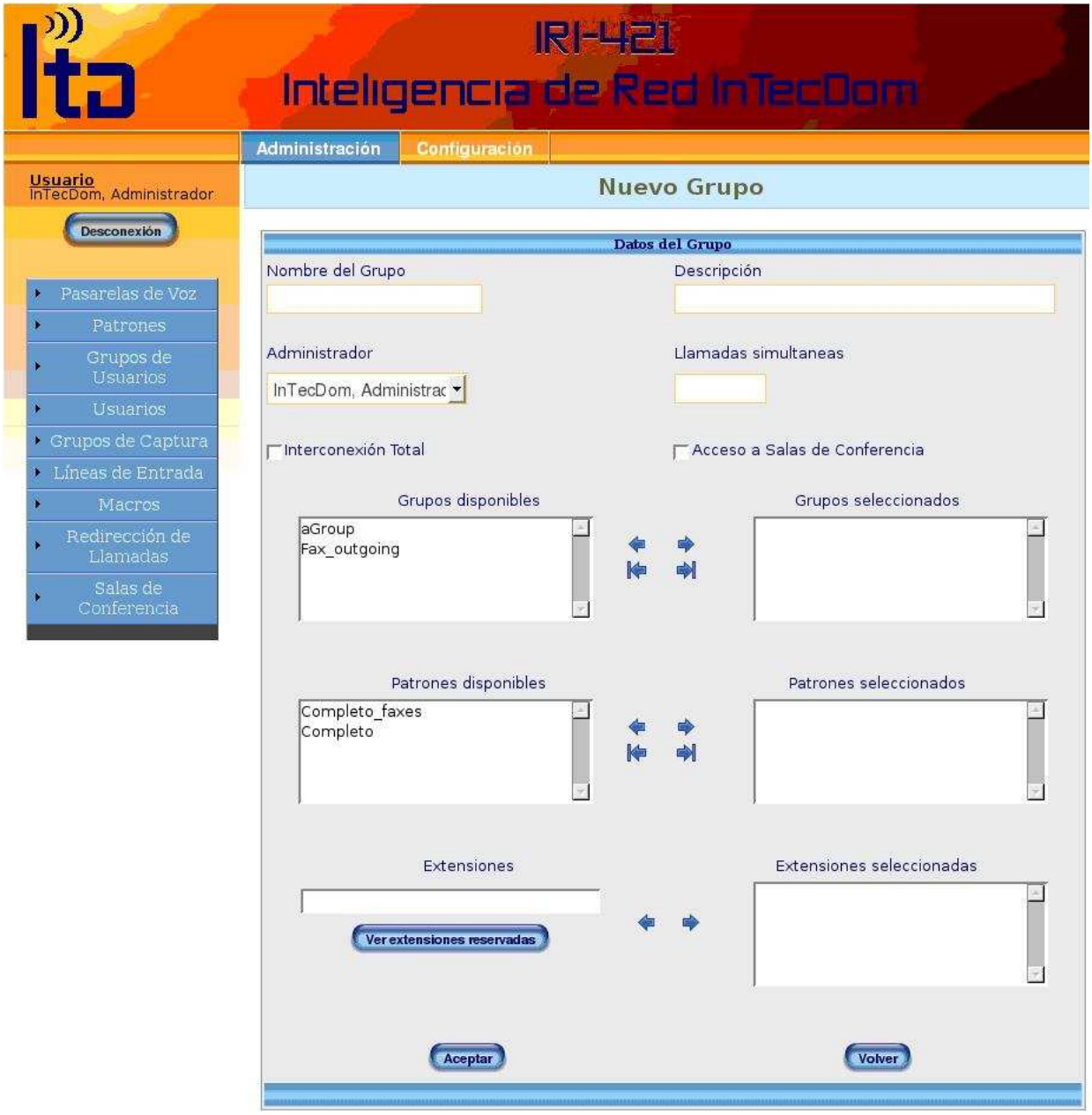
A continuación se detalla cada una de las operaciones posibles.

4.2.1.1. Creación de usuarios en la centralita telefónica

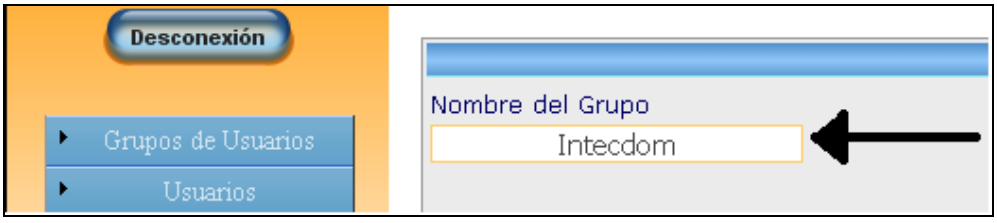
4.2.1.1.1. Creación de grupos de usuarios

Un paso previo a la creación de usuarios es la creación de un grupo al que añadir a ese usuario. Podemos hacerlo accediendo al menú superior, opción de *Administración*:

Al seleccionar la opción *Nuevo Grupo* en el panel izquierdo, se muestra el *Tablero de creación de grupos***,** en el que aparecen los datos a rellenar para crear el nuevo grupo:

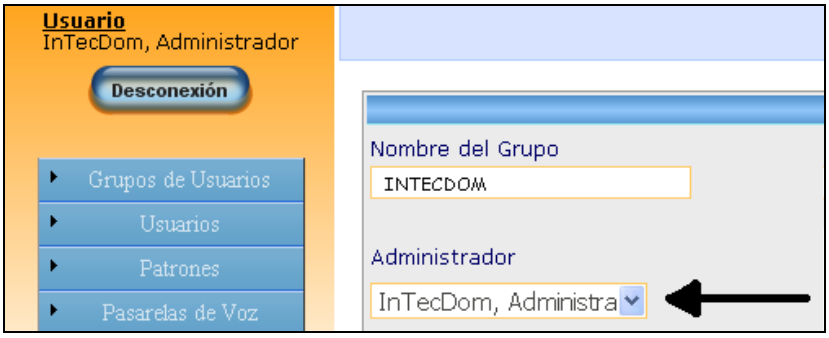


- **Nombre**: En este campo hay que introducir el nombre deseado para el grupo. Por ejemplo: *Intecdom*

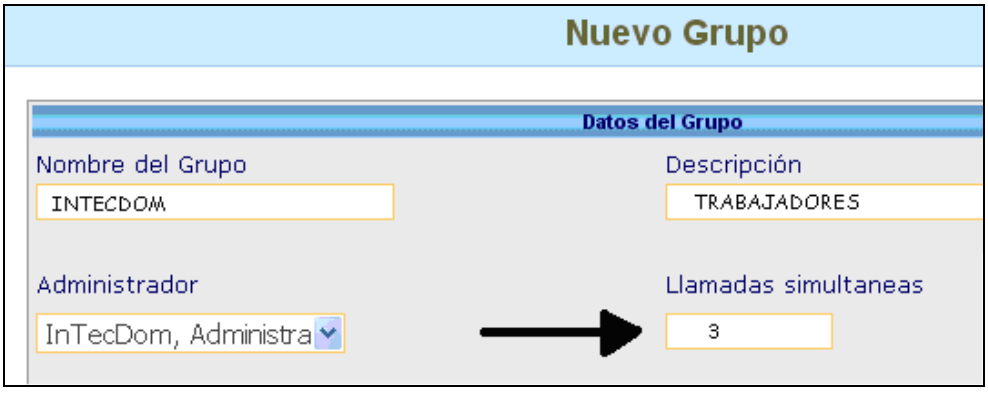


- **Descripción**: Introducir la Descripción del Grupo. Por ejemplo: *Trabajadores*

- **Administrador**: El campo Administrador tiene el valor por defecto del usuario Intecdom, que tiene la responsabilidad de *Administrador*. Sólo puede ser sustituido por otro usuario que también tenga responsabilidad de *Administrador*.

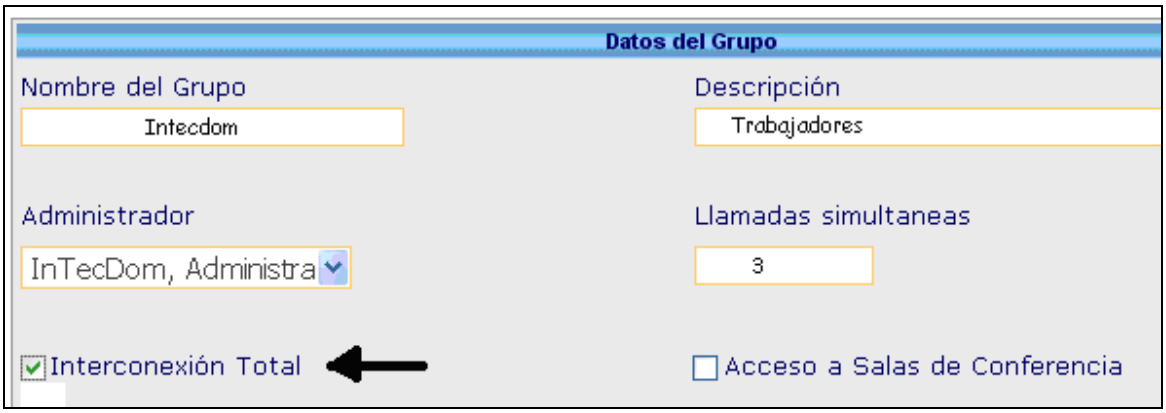


- **Llamadas simultáneas**: Se refiere al número simultáneo de llamadas que pueden tener todos los usuarios de ese grupo.



- **Interconexión Total**: Esta opción se utiliza una vez existen varios grupos definidos. Cuando tenemos varios grupos, podremos definir:
  - Que los grupos se puedan comunicar entre sí
  - Que un grupo pueda comunicarse con el resto de grupos pero no viceversa
  - Que ningún grupo pueda comunicarse entre sí.

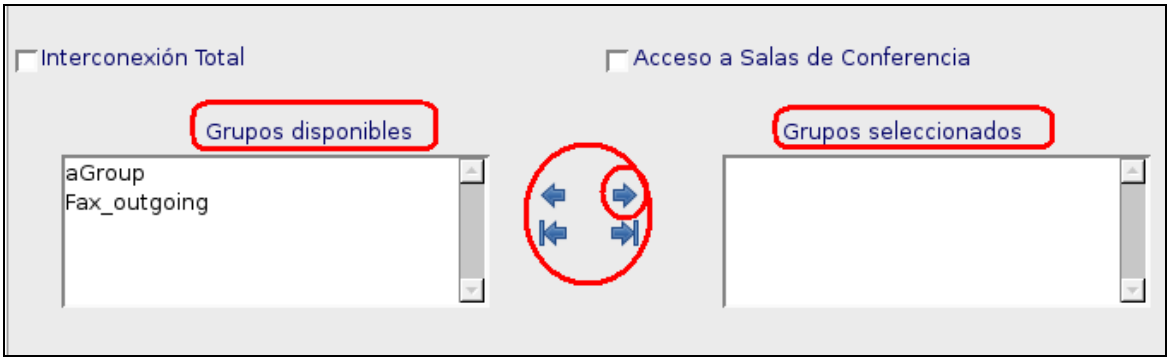
Para tener interconexión entre todos los grupos, hay que seleccionar la opción "*Interconexión total*".



Tras marcar la opción "*Interconexión total*" desaparecen los cuadros de opción "Grupos Disponibles" y "Grupos seleccionados", ya que estos cuadros permiten definir interconexiones concretas entre distintos grupos, opción no compatible con la "*Interconexión total*"

Para que el grupo NO tenga interconexión con ningún otro grupo, no señale la opción "*Interconexión total*" ni seleccione ningún otro grupo de su lista de "*Grupos Disponibles*".

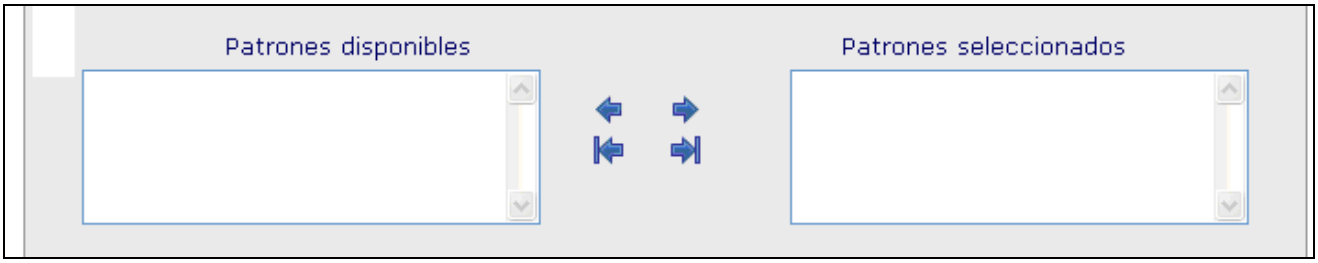
Para que el Grupo sólo tenga interconexión con alguno de los grupos de su lista de "*Grupos disponibles*", señale el grupo o los grupos deseados y haga clic sobre la flecha de la derecha del cuadro para agregarlos a la lista de "*Grupos seleccionados*".



- **Acceso a Salas de Conferencia**: Permite al grupo la utilización de las salas de conferencia.

- **Patrones**: Los patrones permiten restringir o no las llamadas de los grupos. Por ejemplo, para que un grupo pudiera hacer llamadas nacionales habría que seleccionar el patrón necesario para ello. De esta forma se pueden restringir las llamadas a móviles, internacionales o a números 900.

La creación y gestión de los patrones se realiza en la sección correspondiente del menú de administración.

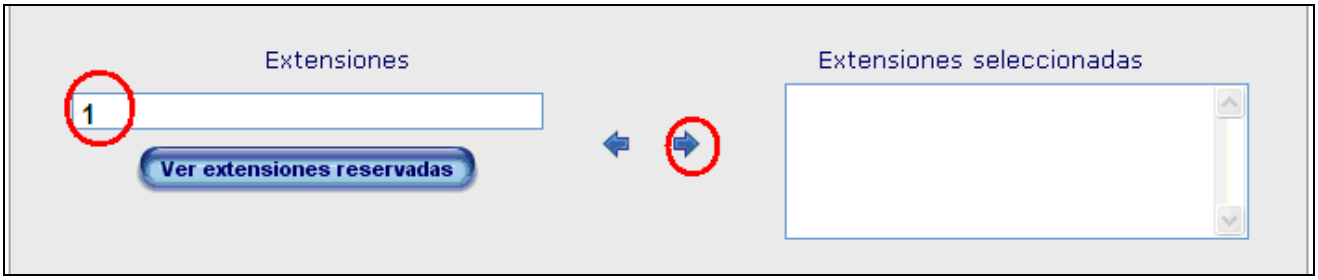


- **Extensiones**: Las extensiones son los números de teléfono que pertenecerán a este grupo.

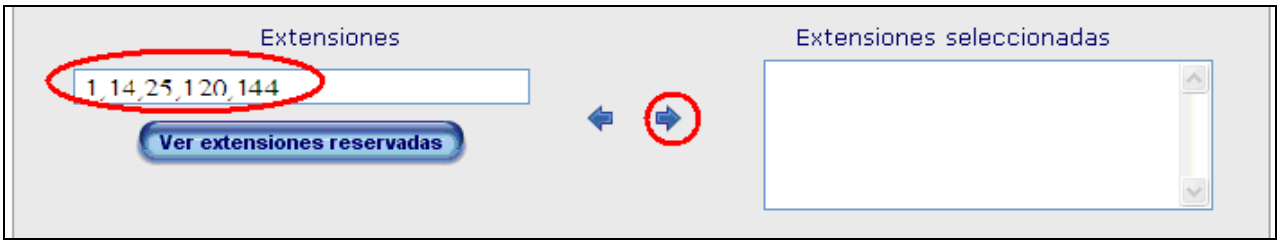
Para asociar las extensiones a usuarios primero hay que "Crear Usuarios", y luego asociarles las extensiones; ver la sección correspondiente del manual, (Pág. **Error! Bookmark not defined.**).

Las extensiones pueden ser introducidas

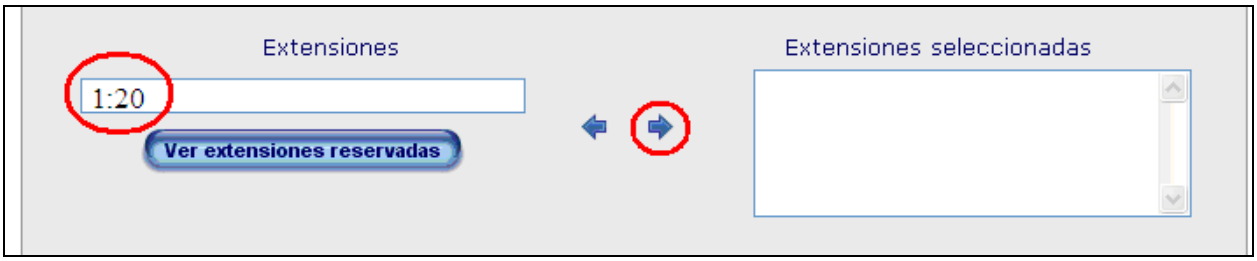
- de una en una, por ejemplo:



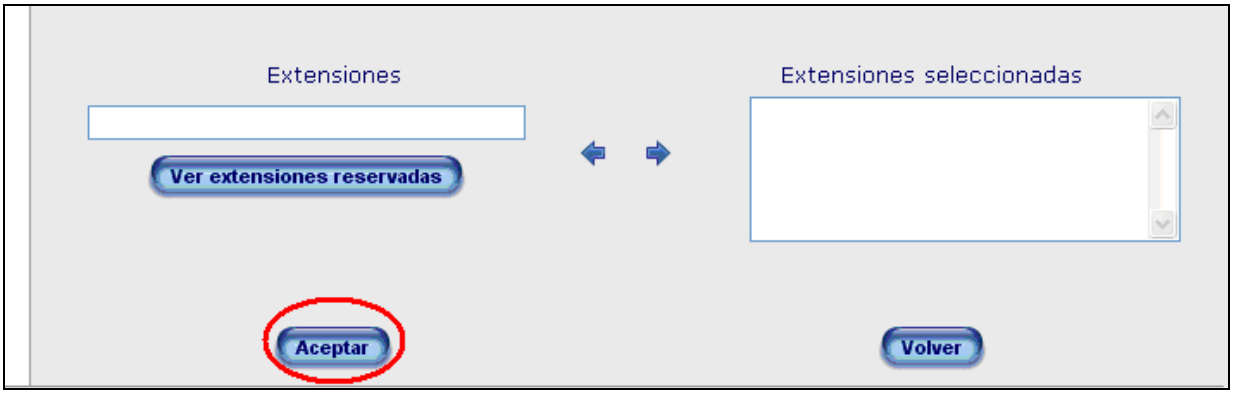
- Varias extensiones no consecutivas, por ejemplo: 1, 14, 25, 120, 144…



- Varios números consecutivos, por ejemplo del 1 al 20 (1:20)



Una vez completados todos los datos deseados para el nuevo grupo, marcar el botón de la parte inferior de la pantalla: “Aceptar”



Tras la creación de al menos un grupo de usuarios, podemos proceder a la creación de usuarios.

4.2.1.1.2. Creación de usuarios. Dos casos: si el usuario aún no existe en LDAP o si el usuario ya existe en LDAP

En este punto vamos a recordar las decisiones de diseño adoptadas previamente referentes a la creación de usuarios:

- Al crear un usuario de telefonía, se incluirá a este usuario en una unidad organizativa que permita identificarlo como usuario de Asterisk, es decir, se creará una ou=*usuariosIRI* que será añadida a los usuarios de telefonía IP y permitirá realizar búsquedas sobre ellos con un sencillo filtro.

Esta decisión afecta al modo en que se crea un nuevo usuario en el directorio LDAP. El usuario pasará a formar parte del árbol de directorio incluido dentro de una unidad organizativa llamada *usuariosIRI* que permitirá su fácil identificación.

- Como los nuevos usuarios de telefonía pueden ser usuarios ya existentes en LDAP o ser usuarios completamente nuevos, el proceso de creación de usuarios se basará en una búsqueda filtrada por nombre y apellido (CN) sobre las entradas del directorio. Si el usuario existe, se le incluye en la clase usuariosIRI y se le añaden los atributos específicos de Asterisk a sus atributos ya existentes, utilizando la opción *Modify* De LDAP. *Modify* precisa del DN de la entrada a modificar, que se obtendrá mediante la operación *LDAPsearch* anterior. Si el usuario es completamente nuevo, se añadirá una entrada en el directorio que implementará las clases *inetOrgPerson* y *usuariosIRI*, y se definirá el DN de la entrada con un UID igual al Nombre de Usuario definido en el formulario Web de recogida de datos de la aplicación.

Esta decisión afecta al modo en que se crea un nuevo usuario en el directorio LDAP, de modo similar a la decisión anterior. En este caso se asume que los nuevos usuarios de telefonía se crearán en el directorio LDAP dotándolos de unas características básicas comunes a todos los elementos de tipo *inetOrgPerson*. Asumir esta decisión no es arriesgado, teniendo en cuenta la naturaleza de los usuarios de nuestro sistema y los atributos característicos de la clase *inetOrgPerson*.

Por otro lado, la utilización de la opción *Modify* de LDAP para ampliar los usuarios de nuestro sistema al grupo de usuarios ya existentes en el directorio, dota de una mayor funcionalidad y hace más práctica esta aplicación.

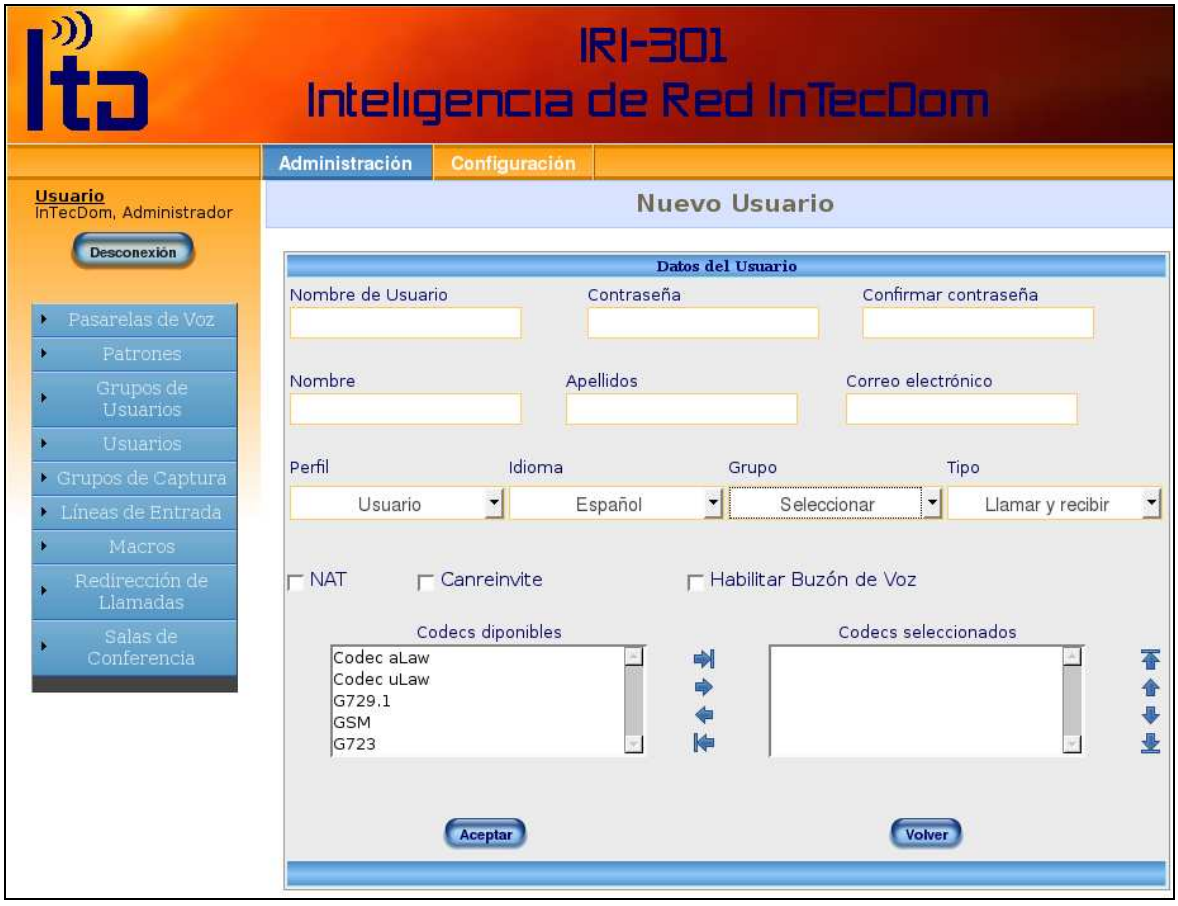
La principal función desarrollada para implementar esta funcionalidad es la función ***newUserLDAP*** descrita en este documento.

En resumen, todos los usuarios se añaden a la centralita telefónica mediante el menú crear usuarios. Si el nombre y apellidos del usuario ya existían el directorio LDAP, esta entrada será modificada para añadirla al grupo de usuarios de la centralita y para añadirle los atributos necesarios para convertirla en un usuario de Asterisk. Si el nombre y apellidos son nuevos, se creará una nueva entrada LDAP con los atributos anteriormente descritos. En ambos casos, se creará un usuario en la base de datos PostgreSQL que contenga las características introducidas en el menú de creación de usuarios para mantener la funcionalidad previa de la IRI.

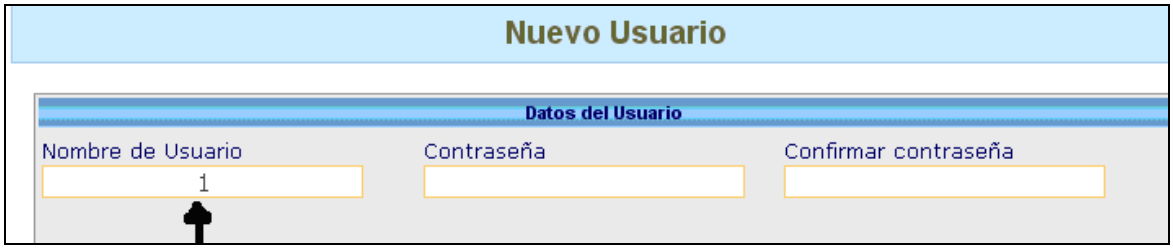
A continuación se adjunta un extracto de las instrucciones del manual de utilización de la centralita telefónica, en los apartados respectivos a la creación de un nuevo usuario y de su extensión telefónica.

Creación de usuarios

Al seleccionar la opción *Nuevo Usuario* en el panel izquierdo, se muestra el Tablero de creación de usuarios, en el que aparecen los datos a rellenar para crear el nuevo usuario:

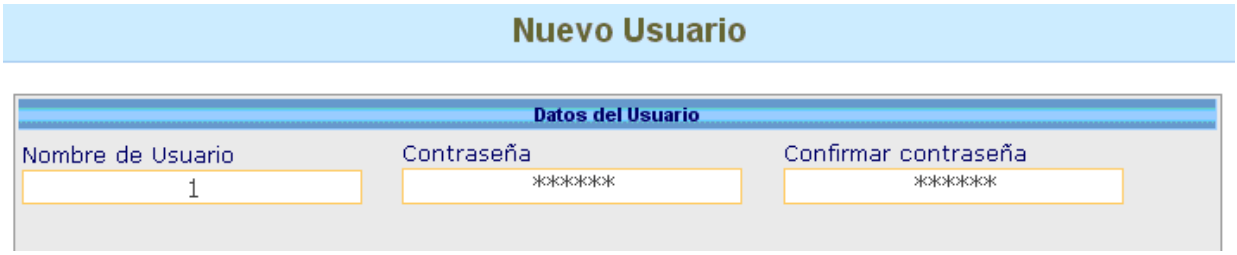


Nombre de Usuario: Introduzca el nombre de Usuario que desea crear. Este nombre identificará al usuario en el sistema (equivalente al UID en LDAP)



Contraseña: Introduzca una contraseña para este usuario. El usuario, desde la página de inicio, introduciendo su nombre y su contraseña podrá acceder al sistema.

Confirmar contraseña: Introducir el mismo valor que en el campo Contraseña.



Nombre: Introduce el nombre propio del Usuario

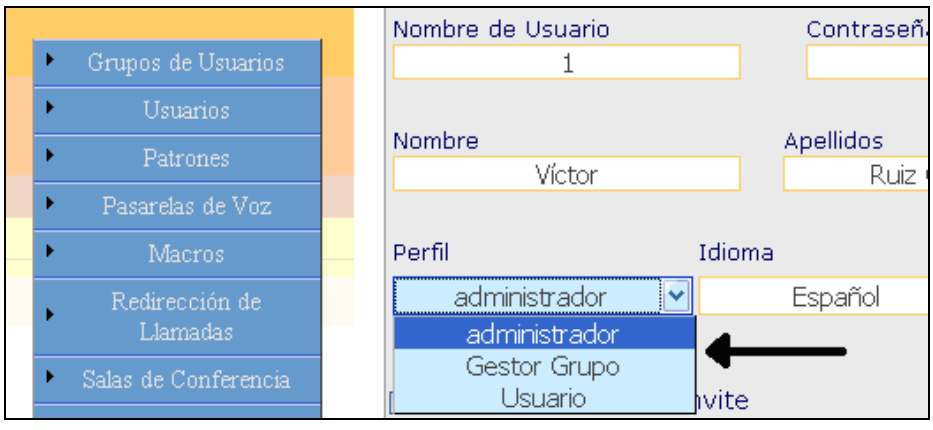
Apellidos: Introduce los Apellidos del Usuario

Nota: el concepto de *Common Name* de LDAP (CN) utilizará la concatenación de los dos campos anteriores, de esta forma: “Nombre_Apellidos”.

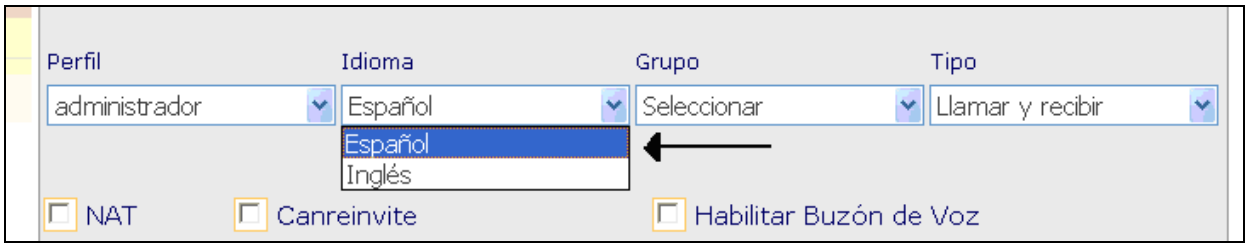
Correo electrónico: Introduce el correo electrónico del Usuario. Este campo se utiliza entre otras cosas para enviar mensajes de Buzón de Voz.

Perfil: Tenemos 3 opciones:

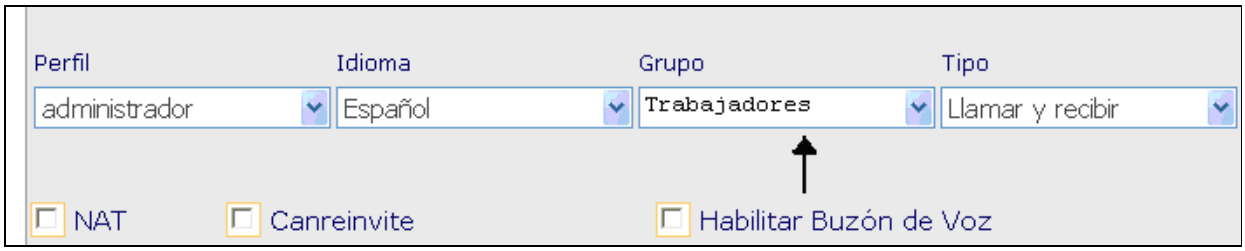
- Administrador, que permite al usuario, una vez creado en el sistema, administrar la IRI, creando usuarios, grupos etc.
- Usuario, que podrá acceder a sus datos y modificarlos dentro del sistema.
- Gestor de Grupos, esta opción se encuentra de momento deshabilitada.



Idioma: Aquí se puede elegir el idioma en el que se expondrá el menú cuando el usuario entre en el sistema.

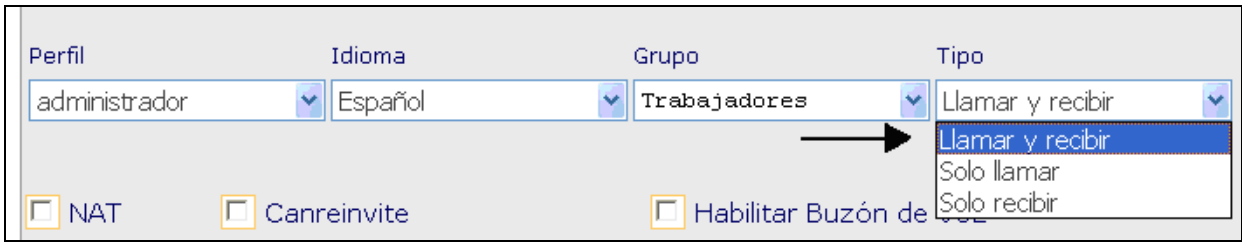


Grupo: Para poder elegir un grupo, antes hay que haber creado alguno, siguiendo las instrucciones previas. En este ejemplo elegiremos el grupo "Trabajadores"



Tipo: Este campo define el tipo de llamadas que están permitidas para este usuario:

- Llamar y recibir: el nuevo usuario podrá llamar y recibir llamadas.
- Llamar: el nuevo usuario sólo podrá realizar llamadas, pero no recibirlas
- Recibir: el nuevo usuario sólo podrá recibir llamadas, pero no realizarlas



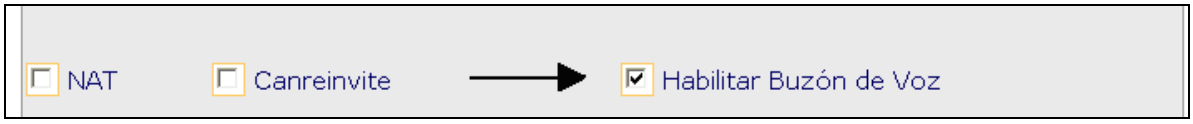
NAT: esta opción hay que marcarla cuando el Usuario utilice un teléfono a través de Internet.

Con esta opción, la centralita asocia a la extensión telefónica una IP pública, que son las IPs válidas en Internet, en lugar de una IP interna o privada.

Canreinvite: esta opción es incompatible la opción de NAT, por lo que se podrá marcar cuando no se haya marcado la opción anterior.

Esta opción permite que las llamadas de usuarios dentro de la LAN no pasen por la IRI, con lo que disminuirá el tráfico en la IRI y aumentará su eficiencia.

Habilitar Buzón de Voz: Seleccione esta opción si desea que el Usuario tenga un buzón de voz.

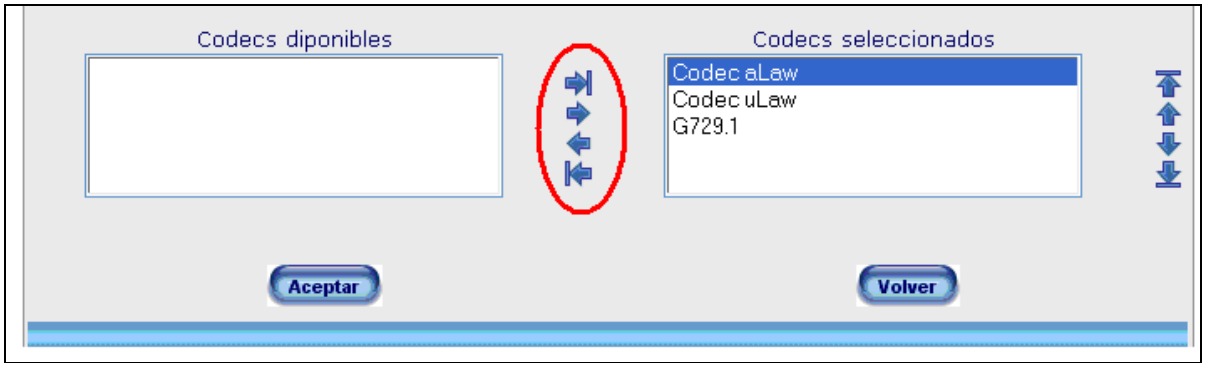


Codecs:

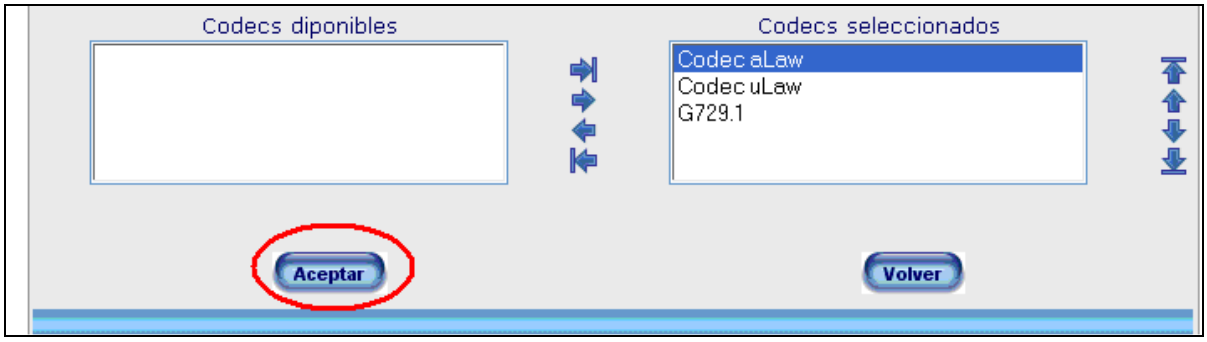
La casilla Codecs Disponibles muestra los codecs del teléfono que están disponibles. Los codecs son los que determinarán la calidad de la voz en los sistemas de telefonía VoIP.

La IRI permite seleccionar todos los codecs, dando prioridad a unos frente a los otros. Esta prioridad está marcada por su posición, es decir, el que se encuentre el primero dentro de la casilla de Codecs Seleccionados tendrá preferencia

Para seleccionar un codec, utilice las flechas azules.



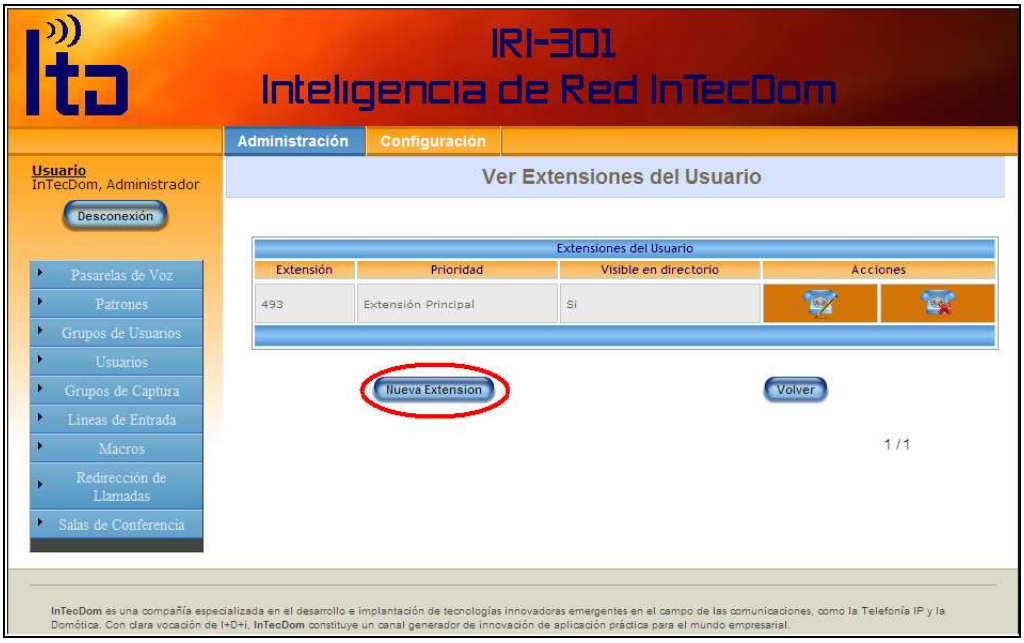
Una vez completados todos los datos deseados para el nuevo usuario, marcar el botón de la parte inferior de la pantalla: "*Aceptar*"



En este punto hemos terminado de rellenar el tablero de creación de usuario, pero para terminar de definir a este usuario aún nos faltaría asociarle una extensión. Para asociar una extensión a un usuario utilizamos la funcionalidad existente en la IRI, que no necesita ser modificada para la integración con LDAP.

Creación de extensiones de usuario

Al seleccionar la opción Ver Extensiones del Usuario en el botón derecho del panel de Acciones, se muestra un listado con las extensiones existentes.



Si esta extensión corresponde a un usuario recién creado, el listado de extensiones estará vacío.

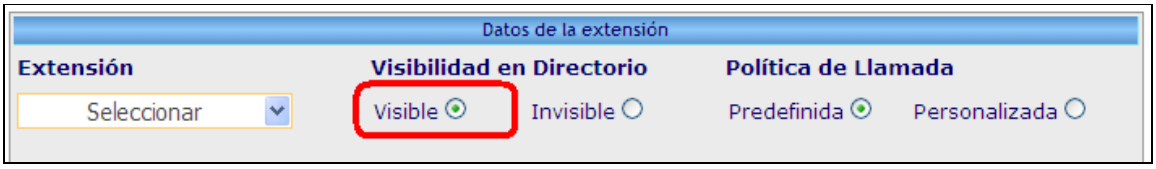
Seleccionando el botón de Nueva Extensión se muestra el Tablero de creación de extensiones, en el que aparecen los datos a rellenar para crear una extensión para el usuario:

Podemos ver las siguientes características de la extensión:

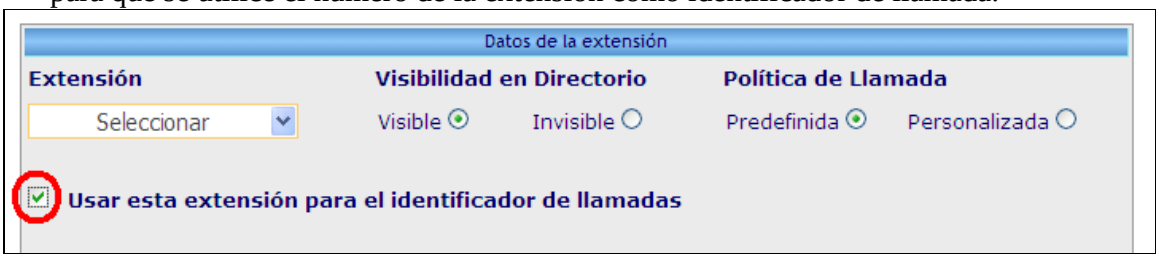
- **Extensión:**

Número correspondiente a la extensión. En el desplegable aparecerán las extensiones que se hayan definido para el grupo al que pertenece el usuario, y que no estén siendo ya utilizadas.

- **Visibilidad en Directorio**: se puede elegir entre *visible* o *invisible*, según queramos que la extensión aparezca o no en el directorio. En este caso seleccionamos *visible*.



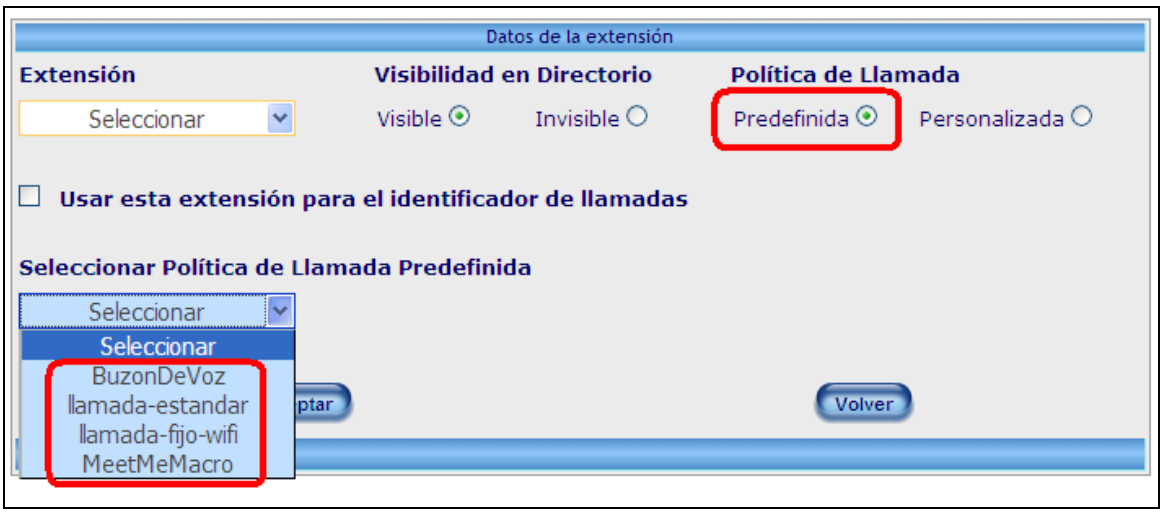
- **Usar esta extensión para el identificador de llamadas:** seleccionaremos esta opción para que se utilice el número de la extensión como identificador de llamada.



- **Política de Llamada:** se puede elegir entre *predefinida* o *personalizada*, según queramos optar por una política de llamada predefinida en las macros, o queramos personalizar esta política mediante la definición de un código Asterisk concreto:

  - Al seleccionar la opción *Predefinida*, tendremos que elegir la macro que nos convenga en el apartado *Seleccionar Política de Llamada Predefinida*.

    Las políticas de llamadas que aparecen en el desplegable son las macros predefinidas que no son de pasarela:
    - Buzón de Voz
    - Llamada-estándar
    - Llamada-fijo-wifi
    - MeetMeMacro



Seleccionando la tecla *Aceptar* terminaremos el registro de la nueva extensión.

4.2.1.2. Modificación de usuarios

Tal y como se comentó en el punto anterior sobre la creación de usuarios, esta aplicación ha sido pensada para poder reutilizar usuarios ya creados en un directorio LDAP existente. Para ello se realiza una búsqueda filtrada por nombre y apellido (CN) sobre las entradas del directorio. Si el usuario existe, se le incluye en la clase *usuariosIRI* y se le añaden los atributos específicos de Asterisk a sus atributos ya existentes, utilizando la opción Modify De LDAP. *Modify* precisa del DN de la entrada a modificar, que se obtendrá mediante la operación *LDAPsearch* anterior.

La principal función desarrollada para implementar esta funcionalidad es la función *modifyUserLDAP* descrita en este documento.

Cualquier modificación de usuarios de la centralita tendrá efecto tanto en el directorio LDAP como en la base de datos interna PostgreSQL, para mantener la sincronización entre ambas.

4.2.1.3. Borrado de usuarios

El borrado de usuarios se realiza mediante la funcionalidad *Delete* de LDAP. Tras introducir las credenciales del usuario que se desea borrar, se procede a la búsqueda de este usuario en el directorio LDAP, y si la búsqueda tiene éxito se procederá a la eliminación del usuario, tanto en la base de datos de la centralita telefónica IRI, como en el directorio LDAP.

Otros escenarios fueron barajados a la hora de definir la funcionalidad de borrado, por ejemplo el borrado únicamente en la base de datos de la centralita telefónica y no en el directorio. Esta opción era más fácil de implementar, ya que el borrado de usuarios ya está implementado en

la IRI, por lo que la función de borrado no necesitaría ser modificada para la nueva funcionalidad de LDAP

Finalmente se optó por la solución de borrado en ambos lugares por dos motivos; el primero es ampliar la funcionalidad de la opción de borrado existente, y el segundo es la apreciación de que la opción de borrado es únicamente accesible desde la centralita telefónica mediante el usuario interesado y el usuario administrador, considerándose ambos usuarios con derecho a borrado. De esta forma además se mantiene la sincronización entre la información de usuarios de telefonía en el directorio LDAP y en la base de datos interna PostgreSQL.

La principal función desarrollada para implementar esta funcionalidad es la función *deleteUserLDAP* descrita en este documento.

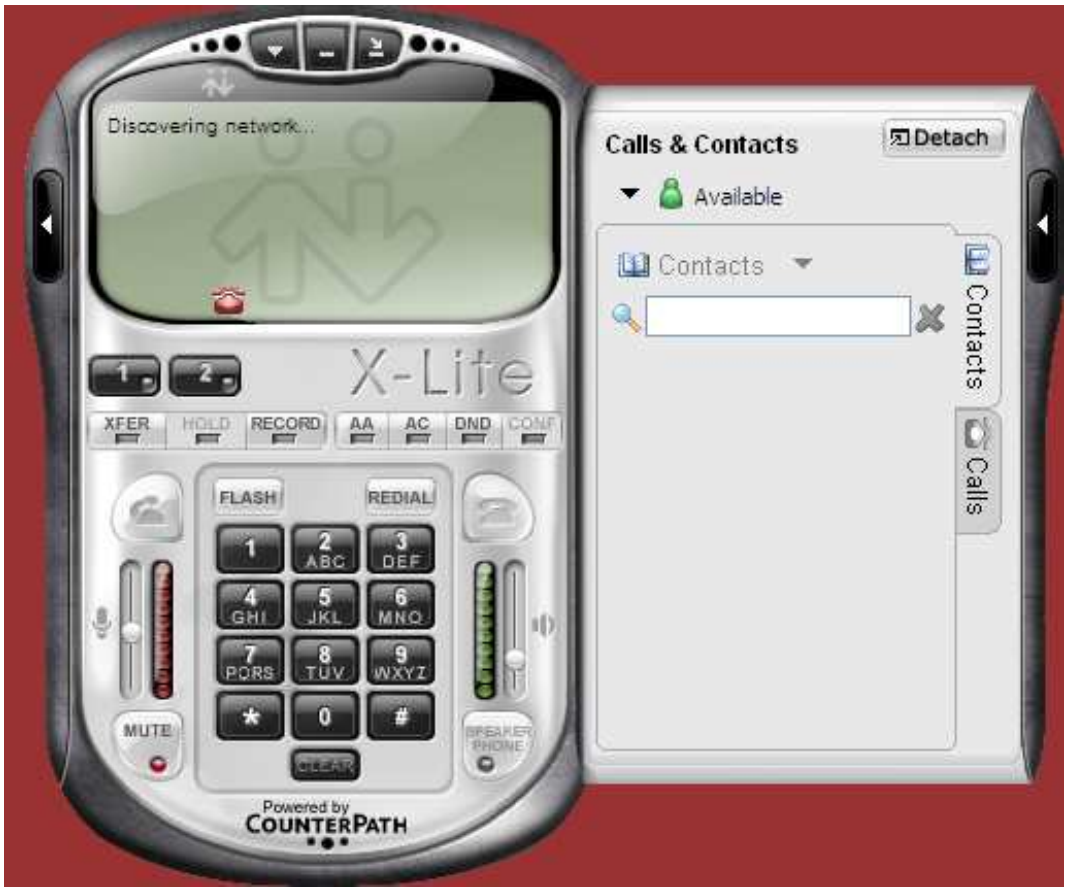
4.2.2. Autenticación en el inicio de llamadas

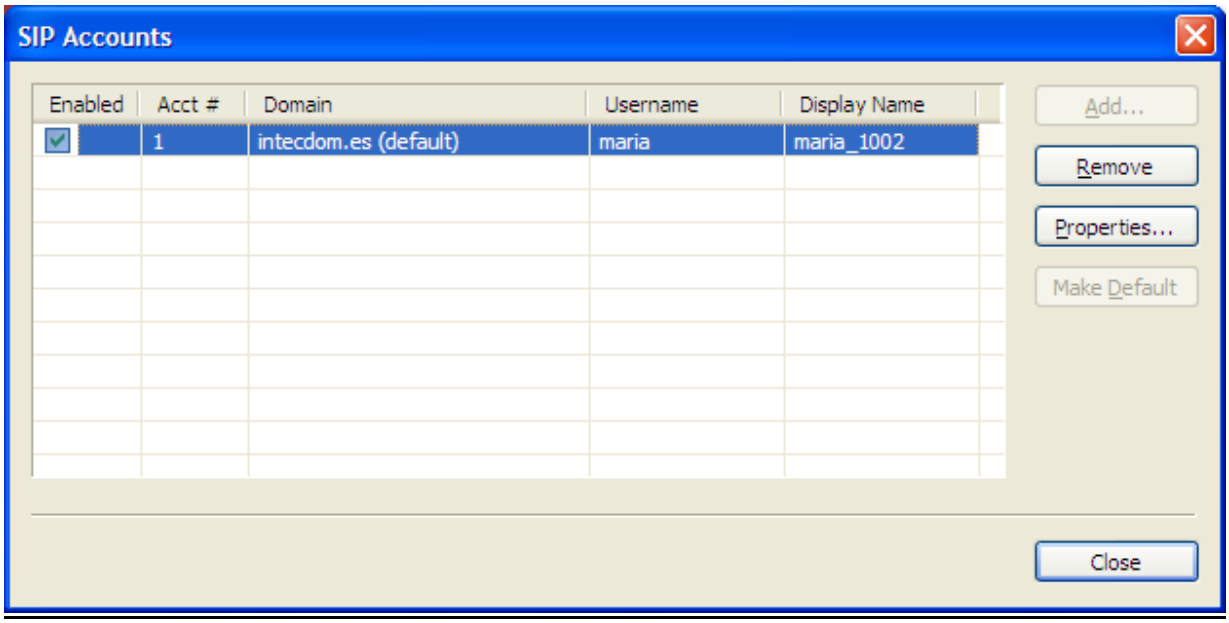
Una de las ventajas de las nuevas librerías que integran LDAP con Asterisk es la utilización directa de los usuarios de LDAP que contienen ciertos atributos de Asterisk como usuarios SIP, sin necesidad de validarlos por medio del fichero SIP.conf como venía siendo habitual en las versiones anteriores de Asterisk.

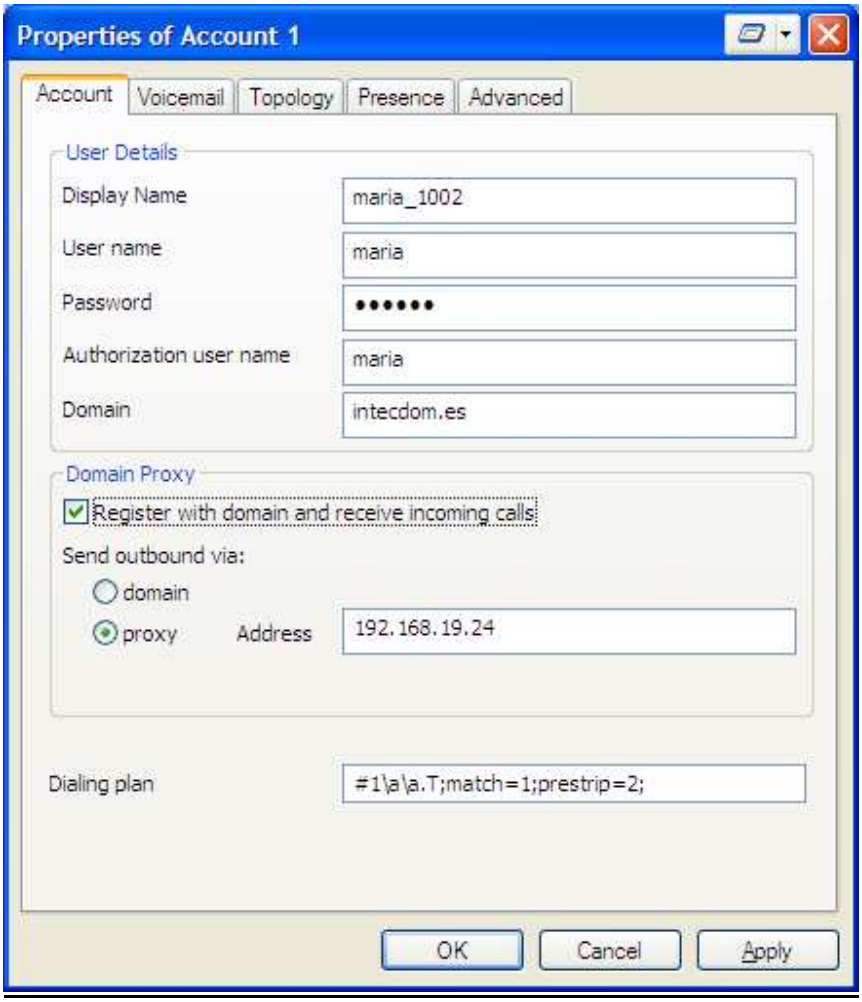
La indicación de este modo de funcionamiento mediante LDAP en lugar de mediante ficheros de configuración, se hace mediante las definiciones en los ficheros de configuración /etc/Asterisk/extconfig.conf, que es el que indica desde dónde debe Asterisk leer sus configuraciones, y desde /etc/Asterisk/res_ldap.conf, que configura el comportamiento del *backend* LDAP de Asterisk.

Se pudo comprobar cómo al iniciarse una llamada la búsqueda del usuario llamante pasa a hacerse en el directorio LDAP en lugar de en el fichero de configuración SIP.conf, tal como indica el fichero extconfig.conf.

Por ejemplo, podemos observar en las siguientes figuras uno de los *softphones* utilizados en la batería de pruebas, y su configuración de usuario:


Discovering network...
X-Lite
XFER
HOLD
RECORD
AA
AC
DND
CONF
FLASH
REDIAL
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
*
0
#
MUTE
SPEAKER PHONE
CLEAR
Powered by
COUNTERPATH
Calls & Contacts
Detach
Available
Contacts
Contacts
Calls


SIP Accounts
Enabled
Acct #
Domain
Username
Display Name
1
intecdom.es (default)
maria
maria_1002
Add...
Remove
Properties...
Make Default
Close



Este usuario fue configurado inicialmente mediante la aplicación IRI en el menú de creación de usuarios, luego se verificó su creación en el directorio LDAP, posteriormente se fijaron los atributos correspondientes en la configuración del *softphone* X-LITE y finalmente se realizaron llamadas entre este usuario y otros usuarios registrados en la centralita IRI.

5.MEMORIA ECONÓMICA Y PRESUPUESTO.

En este capitulo se realizará un estudio estimativo de los costes de ejecución del proyecto descrito. Estos costes serán calculados mediante el formulario de presupuestos estándar propuesto para proyectos de la UC3M.

Como comentario a los costes de personal incluidos en el presupuesto, me he basado en una dedicación por mi parte de un mes a tiempo completo, y 6 meses a media jornada. La dedicación por parte del tutor en empresa la he estimado en 3 días al mes, y la dedicación del tutor en la universidad en un día al mes.

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Maria Ana Ruiz Cañamero

2.- Departamento: Ing. Telemática

3.- Descripción del Proyecto:

- Titulo: **Autenticación y administración centralizada en sistemas VoIP con Asterisk y LDAP**
- Duración (meses): **7**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

11.954,22 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Ruiz, Jose Manuel		Ingeniero Senior	0,18	4.289,54	772,12	
Muñoz, Mario		Ingeniero Senior	0,06	4.289,54	257,37	
Ruiz Cañamero, Maria Ana		Ingeniero	4	2.694,39	10.777,56	
					0,00	
					0,00	
		Hombres mes	**4,24**	**Total**	11.807,05	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
PC con SO Win. XP y MS Office	700,00	70	7	60	57,17
PC con SO Debian	500,00	100	7	60	58,33
Servidor de Intecdom	1.000,00	20	6	60	20,00
Telefono IP Hardware	100,00	100	7	60	11,67
Telefonos IP Software	0,00	100	7	60	0,00
					0,00
				Total	147,17

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	11.807
Amortización	147
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	2.391
Total	14.345

6.CONCLUSIONES Y POSIBLES MEJORAS.

En este capítulo se exponen las conclusiones obtenidas en el presente proyecto, para concluir aportando algunas mejoras que podrían ser implementas en versiones futuras de este proyecto.

6.1. Conclusiones

He aprendido muchas cosas durante la planificación, desarrollo y documentación de este proyecto: LDAP, Asterisk, integración de tecnologías, programar en Java y utilizar Eclipse y Maven, entre otras.

Globalmente se pueden señalar como ventajas de la integración desarrollada el hecho de estar basada en software libre, que facilita un bajo coste de desarrollo, así como la utilización de tecnología VoIP para las comunicaciones, que aprovecha excedentes en la red de datos evitando costes adicionales en líneas telefónicas dedicadas.

Mi experiencia durante la realización del proyecto es que todas las tecnologías implicadas experimentan un rápido desarrollo, lo cual hace necesaria una continua evolución para adaptarse a las nuevas versiones ofrecidas. El esfuerzo que implican estas actualizaciones es grande, lo cual hace que hace que sea difícil mantener un producto que integre la lúltima versión de todas y cada una de ellas.

Del mismo modo he aprendido que un proyecto de integración requiere más esfuerzo de lo que en principio pueda parecer, y es más complicado cuantos más elementos distintos lo compongan, ya que los elementos que influyen a la hora de testear y trazar errores, los cuales pueden provenir de diversas fuentes, por lo que se hace necesaria una planificación del testeo y del depurado de código detallada y por fases que facilite la identificación y verificación de errores.

Con respecto al estado de OpenLDAP, éste es un campo en pleno desarrollo, con diversas implantaciones que van abriéndose camino en diferentes entornos. Una fuente de ideas, soluciones y puesta en común ha sido para mí el hecho de unirme a un foro en línea de usuarios de OpenLDAP, donde cada día se recibe una media de 10 correos electrónicos de gente que expone sus problemas, preguntas y logros para compartir con los demás usuarios, lo cual nos puede dar una idea de la actividad de esta comunidad.

Sobre el desarrollo y futuro de la integración de LDAP con Asterisk, he de decir que después de cierta documentación encontrada sobre tal integración realizada por una empresa vasca a finales de 2006, no he encontrado mucha más información relevante. Pero el desarrollo de los módulos de integración LDAP en Asterisk, así como el esquema de Asterisk para LDAP, muestran inequívocamente una tendencia a la consolidación y estandarización de la integración de ambas tecnologías.

En resumen, integrar un sistema de telefonía y un directorio de usuarios tiene un importante interés comercial, más aún tratándose de la mejora de un producto ya comercializado en el mercado, para el cual se introduce la posibilidad de gestionar la autenticación mediante directorio LDAP.

6.2. Mejoras futuras

Por último, se marcarán las pautas ante una posible continuidad del proyecto y se presentarán mejoras sobre la versión actual.

A continuación algunas posibles mejoras identificadas durante la realización del proyecto:

- Incluir cifrado con TLS en las comunicaciones entre la centralita y el directorio.
- Investigar la posibilidad de incluir todas las dependencias de Asterisk en LDAP en lugar de en los ficheros de Asterisk
- Incluir replicación de Asterisk, para aumentar la seguridad ante contingencias.
- Incluir replicación de LDAP, para aumentar la seguridad ante contingencias.
- Investigar la posibilidad de hacer un volcado inicial de todos los usuarios existentes en LDAP a la IRI, analizando el tema de la creación de contextos de llamada previamente y cómo podrían asignarse los distintos contextos a distintos grupos de usuarios, si hubiera posibilidad de automatizarlo.

Una posible evolución sería realizar un producto nuevo que considerase como base de autenticación un directorio LDAP existente, en lugar de reutilizar un desarrollo existente en el que se paralelice el uso de una base de datos local con el directorio externo.

7.BIBLIOGRAFÍA Y REFERENCIAS

LDAP:

Bibliografía:

- Understanding and deploying LDAP directory services. Howes, Timothy A. (Macmillan Network Architecture and Development Series)
- LDAP programming with Java. Weltman, Rob
- Implementing LDAP. Wilcox, Mark
- Desarrollo de una herramienta de migración para directorios LDAP del servidor AAA. Cano Blázquez, Miguel Ángel

Principales Referencias:

- http://www.ietf.org
- http://www.rfc-editor.org/rfc/rfc4510.txt
- http://www.openldap.org/
- http://www.debian-administration.org/article/OpenLDAP_installation_on_Debian
- http://www.voip-info.org/wiki/view/LDAP

Java:

Bibliografía:

- Profesional programación Java Server con J2EE Edición 1.3. Allamaraju, Subrahmanyam

Principales Referencias:

- http://java.sun.com/docs/books/tutorial/

Asterisk:

Bibliografía:

- Asterisk. The Future of Telephony. Jim Van Meggelen, Pared Smith, Leif Madsen. O'Reilly. 2005.
- Sergio Serrano. Asterisk en Español. Astricon Europe. 2005

- Herramienta para la configuración de una Centralita SIP Asterisk. Javier Vendrell García

Principales Referencias:

- www.asterisk.org

- www.voipforo.com

Eclipse:

Principales Referencias:

- www.eclipse.org

- http://eclipsetutorial.forge.os4os.org/proyecto.htm

Maven:

Principales Referencias:

- http://maven.apache.org/
- http://es.debugmodeon.com/articulo/maven-guia-rapida

Otros enlaces útiles:

- www.wikipedia.org

- www.opensource.org

8.ACRÓNIMOS

Término	Definición
API	Application Program Interface
ACL	Access Control List
BGP	Border Gateway Protocol. Protocolo de enrutado básico de Internet
CN	Common Name
CPU	Central Processing Unit
DAP	Directory Access Protocol
DAO	Data Access Object
DC	Domain Component
DIT	Directory Information Tree
DNS	Domain Name System
DN	Distinguished Name
DNS	Domain Name Server
DUA	Directory User Agent
ECJ	Edifact Java Compiler
EJB	Enterprise Java Beans
FAQ	Frequently Asked Questions
FTP	File Transfer Protocol
IAX	Inter-Asterisk Exchange Protocol
IDN	Internationalized Domain Name
ID	Identifier
IDE	Integrated Development Environment
IP	Internet Protocol
ISO	International Standards Organization
IVR	Interactive Voice Response
J2EE	Java Platform Enterprise Edition
JDT	Java Development Toolkit
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
LAN	Large Area Network
LDAP	Lightweight Directory Access Protocol
LDAP Sync	LDAP Content Synchronization
LDAPv3	LDAP, version 3
LDIF	LDAP Data Interchange Format
LTS	Long Time Support
MD5	Message Digest 5
NAT	Network Address Translation

NSS	Name Service Switch. NSS is the service that maps attributes to names.
OID	Object Identifier
OSI	Open Systems Interconnect
PLA	PHP LDAP Admin
PBX	Private Branch eXchange. Siglas utilizadas para designar las Centralitas Telefónicas.
PAM	Pluggable Authentication Modules. Is the method by which FreeBSD authenticates most of its sessions
POM	(Maven) Project Object Manager file (pom.xml)
POSIX	Portable Operating System Interface
RDN	Relative Distinguished Name
RFC	Request for Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman, los primeros que publicaron la descripción del algoritmo de encriptación de clave pública
RTP	Real-time Transport Protocol
SASL	Simple Authentication and Security Layer
SDP	Session Description Protocol
SHA1	Secure Hash Algorithm 1
SLAPD	Standalone LDAP Daemon
SLURPD	Standalone LDAP Update Replication Daemon
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer. A communications protocol, it is considered deprecated. Now TLS is used.
syncrepl	LDAP Sync-based Replication
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UID	User Identifier
URL	Uniform Resource Locator
VoIP	Voice over IP
VPN	Virtual Private Network, o Red Privada Virtual.
WWW	World Wide Web
X.500	X.500 Directory Services
XML	Extensible Markup Language

Nota: La mayoría de acrónimos referentes a LDAP han sido recogidos de la página Web: http://www.openldap.org/doc/admin24/glossary.html#Terms

ANEXOS

1. Código de los nuevos métodos desarrollados

En el contenedor IRI-Beans:

UsersManagerBean.java:

Definición de las funciones:

- getConexionLDAP
- newConexionLDAP

```
package asModule;

import …

public abstract class UsersManagerBean implements javax.ejb.SessionBean {

protected SessionContext ctx;

public void setSessionContext(SessionContext ctx) throws EJBException, RemoteException {
	this.ctx = ctx;
}

…

//Para LDAP
/**
 * @ejb.interface-method view-type="remote"
 * @dao.call name="getConexionLDAP"
 **/
public HashMap getConexionLDAP () throws EJBException {
	return null;
}

/**
 * @ejb.interface-method view-type="remote"
 * @dao.call name="newConexionLDAP"
 **/
public String newConexionLDAP(HashMap conexionLDAP) throws EJBException {
	return null;
}
//

}
```

UsersManagerDAOImpl.java:

Implemantación de las funciones:

- getConexionLDAP
- newConexionLDAP

```
package ejbsrc.dbAccess;

import …

public class UsersManagerDAOImpl implements UsersManagerDAO {

private DataSource iriJDBCFactory;

private Boolean debug = true;

…
//***************metodos LDAP************************//
public HashMap getConexionLDAP () throws EJBException {

	System.out.println("[UsersManagerDAOImpl] Starting method getUserData");
	Connection connIRI = null;
	PreparedStatement psIRI = null;
	ResultSet rsIRI = null;
	HashMap result = null;
	String queryString = "";

	try {
		connIRI = this.iriJDBCFactory.getConnection();
		queryString =  "select * from ldap_conexion" ;
		psIRI=connIRI.prepareStatement(queryString);
		rsIRI = psIRI.executeQuery();

		if (!rsIRI.next()) {
			throw new SQLException("Can't get User data information");
		} else {
			result = new HashMap();
			result.put("ldapHost", rsIRI.getString("ldap_host"));
			result.put("ldapPort", rsIRI.getString("ldap_port"));
			result.put("ldapDomain", rsIRI.getString("ldap_domain"));
			result.put("cn", rsIRI.getString("cn"));
			result.put("password_cn", rsIRI.getString("password_cn"));
		}

	} catch (SQLException e) {
		if (debug)
		System.out.println("[ERROR => UsersManagerDAOImpl.getUserData] Cannot prepare SQL
query");
		e.printStackTrace();
		result = null;
	} finally {
		try {
			if (rsIRI != null) {
				rsIRI.close();
			}
			if (psIRI != null) {
				psIRI.close();
			}
```

```
                    if (connIRI != null) {
                          connIRI.close();
                    }
             } catch (Exception e) {
                    if (debug)
                          System.out.println("[ERROR => UsersManagerDAOImpl.getUserData]
Cannot close database connection: ");
                    e.printStackTrace();
                    result = null;
             }
       }
       if (result != null) {
       }
       return result;
}


public String newConexionLDAP(HashMap conexionLDAP) throws EJBException {

       System.out.println("[UsersManagerDAOImpl] Starting method newUser");
       Connection connIRI = null;
       PreparedStatement psIRI = null;
       ResultSet rsIRI = null;
       String result = "success";

       try {
             connIRI = this.iriJDBCFactory.getConnection();
             String queryString = "select * from ldap_conexion";
             psIRI = connIRI.prepareStatement(queryString);
             rsIRI = psIRI.executeQuery();
             if (!rsIRI.next()) {
                    queryString = "insert into ldap_conexion
(ldap_host,ldap_port,ldap_domain,cn,password_cn) values(?,?,?,?,?);";
                    psIRI = connIRI.prepareStatement(queryString);

                    psIRI.setString(1, conexionLDAP.get("ldapHost").toString());
                    if(conexionLDAP.get("ldapPort").toString().equals("DEFAULT_PORT")){
                          psIRI.setString(2, "389");
                    }
                    else{
                          psIRI.setString(2, conexionLDAP.get("ldapPort").toString());
                    }
                    psIRI.setString(3, conexionLDAP.get("ldapDomain").toString());
                    psIRI.setString(4, conexionLDAP.get("cn").toString());
                    psIRI.setString(5, conexionLDAP.get("password_cn").toString());
                    psIRI.execute();

             }
             else{
                    queryString = "update ldap_conexion set ldap_host=?, ldap_port=?
,ldap_domain=? ,cn=? ,password_cn=? where ldap_host=? ;";
                    psIRI = connIRI.prepareStatement(queryString);

                    psIRI.setString(1, conexionLDAP.get("ldapHost").toString());
                    if(conexionLDAP.get("ldapPort").toString().equals("DEFAULT_PORT")){
                          psIRI.setString(2, "389");
                    }
                    else{
                          psIRI.setString(2, conexionLDAP.get("ldapPort").toString());
                    }
                    psIRI.setString(3, conexionLDAP.get("ldapDomain").toString());
                    psIRI.setString(4, conexionLDAP.get("cn").toString());
                    psIRI.setString(5, conexionLDAP.get("password_cn").toString());
```

```
                    psIRI.setString(6, rsIRI.getString("ldap_host"));
                    psIRI.execute();

             }



      } catch (SQLException e) {
             e.printStackTrace();
             result = "error";

      } finally {
             try {
                    if (rsIRI != null) {
                           rsIRI.close();
                    }
                    if (psIRI != null) {
                           psIRI.close();
                    }
                    if (connIRI != null) {
                           connIRI.close();
                    }
             } catch (Exception e) {
                    if (debug)
                           System.out.println("[ERROR => UsersManagerDAOImpl.newUser] Cannot
close database connection: ");
                    e.printStackTrace();
                    result = "error";
             }
      }
      if (debug)
             System.out.println("[UsersManagerDAOImpl] Leaving method newUser");
      return result;
}
//***************FIN METODOS LDAP*****************//
```

En el contenedor IRI-Web:

UIMenu.java:

En este fichero se describe el código correspondiente al menú de la izquierda visible cuando se accede a la interfaz web mediante el usuario administrador.

Podemos observar el código correspondiente a los tres submenús: Asterisk, Hardware y LDAP.

Estos tres submenús contienten opciones de configuración a las que tan sólo un usuario con responsabilidad de administrador debe tener acceso. Dichas opciones de configuración corresponden a parámetros tales como puerto de escucha de Asterisk, configuración del rango de los puertos RTP, definición de redes IP locales, deteccdión de nuevo hardware instalado (Tarjetas DAHDI o mISDN) etc.

En la siguiente imagen podemos ver la nueva pantalla de configuración de LDAP que ha sido añadida para la autenticación por LDAP

Código relevante de **UIMenu.java:**

```
package iri.jsfComponents.menu;

import …

public class UIMenu extends UIOutput {
Locale locale = null;
ResourceBundle bundle = null;
HashMap userData = null;
private asModule.UsersManager usersBean = null;
private asModule.ProfileManager profilesBean = null;

public void encodeBegin(FacesContext context) throws IOException {
…
// MENU CONFIGURACION DEL SISTEMA (SOLO PARA EL ADMINISTRADOR DEL SISTEMA)
if (services != null && services.contains("adminService") && getAttributes().get("menuType")
== "sysconfig") {
       this.writeAuthMenu(writer);
       this.writeStartMenu(writer);
       this.writeMenuOption(writer, "p7ABt1_1", getBundleString("menuAdminAsterisk"));
       this.writeStartSubOption(writer, "p7ABw1_1", "p7ABc1_1");
       this.writeSubOption(writer, "/iri/SystemConfig/Asterisk/editSIP.jsf",
getBundleString("menuAdminEditSip"));
       this.writeSubOption(writer, "/iri/SystemConfig/Asterisk/editRTP.jsf",
getBundleString("menuAdminEditRtp"));
       this.writeSubOption(writer, "/iri/SystemConfig/Asterisk/editIAX.jsf",
getBundleString("menuAdminEditIax"));
       this.writeSubOption(writer, "/iri/SystemConfig/Asterisk/editCHAN_DAHDI.jsf",
getBundleString("menuAdminEditChan_Dahdi"));
       this.writeEndSubOption(writer);
       this.writeMenuOption(writer, "p7ABt1_2", getBundleString("menuAdminHardware"));
       this.writeStartSubOption(writer, "p7ABw1_2", "p7ABc1_2");
       this.writeSubOption(writer, "/iri/SystemConfig/Hardware/listHardware.jsf",
getBundleString("menuAdminListHardware"));
       this.writeSubOption(writer, "/iri/SystemConfig/Hardware/discoverHardware.jsf",
getBundleString("menuAdminDiscoverHardware"));
       this.writeEndSubOption(writer);

           //para LDAP
       this.writeMenuOption(writer, "p7ABt1_3", getBundleString("menuAdminLDAP"));
       this.writeStartSubOption(writer, "p7ABw1_3", "p7ABc1_3");
       this.writeSubOption(writer, "/iri/SystemConfig/LDAP/LDAPconf.jsf",
getBundleString("menuAdminEditLDAP"));
       this.writeEndSubOption(writer);
       //*********

           this.writeMenuInit(writer, "startMenu");
       this.writeEndMenu(writer);
       }
…
}
…
}
```

UsersServerBean.java:

Este es el fichero que contiene las principales funciones desarrolladas:

- deleteUserLDAP: dado el dn de una entrada a borrar, conecta al directorio, la busca y la borra.
- newUserLDAP: creamos una nueva entrada en LDAP
- modifyUserLDAP: modificamos una entrada de LDAP

Funciones auxiliares:

- existUserLDAP: comprobamos si existe una entrada en LDAP
- search_dn_UserLDAP_by_cn: a partir del cn de una entrada devuelve el dn de esa entrada
- search_pw_UserLDAP_by_cn: a partir del cn de una entrada devuelve la contraseña de esa entrada
- configLDAP: devuelve una conexión LDAP con los parámetros definidos en la entrada
- getConexionLDAP: devuelve un Hashmap con los parámetros de la conexión LDAP

Se incluyen las llamadas *import* a las librerías relacionadas con LDAP

```
package iri.serverBeans;

import …

import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.ldap.*;
import javax.naming.directory.Attribute;
import javax.naming.directory.Attributes;
import javax.naming.directory.BasicAttribute;
import javax.naming.directory.BasicAttributes;
import javax.naming.directory.DirContext;
import javax.naming.directory.InitialDirContext;

import com.novell.ldap.LDAPAttribute;
import com.novell.ldap.LDAPAttributeSet;
import com.novell.ldap.LDAPConnection;
import com.novell.ldap.LDAPEntry;
import com.novell.ldap.LDAPException;
import com.novell.ldap.LDAPModification;
import com.novell.ldap.LDAPObjectClassSchema;
import com.novell.ldap.LDAPSchema;
import com.novell.ldap.LDAPSearchResults;

public class UsersServerBean implements Serializable {
…
private asModule.UsersManager userEJB = null;

public class UsersServerBeanException extends Exception {
       UsersServerBeanException(String message) {
              super(message);
       }
}

…
```

```
//***************metodos proyecto LDAP******************//

/** deleteUserLDAP
  * @param userData: HashMap que contiene los datos de la entrada LDAP
  * @param conexión_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: No devuelve ningun valor.
  * @throws: popupLDAPDeleteUserError: No se ha podido eliminar el usuario en LDAP
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  */
public void deleteUserLDAP(HashMap userData, HashMap conexion_LDAP)throws
UsersServerBeanException{
       JSONArray errorList = new JSONArray();
       String result = "";

       try {
         LDAPConnection lc = new LDAPConnection();
         UsersServerBean userBean=new UsersServerBean();
         //busco dn
         String dn=userBean.search_dn_UserLDAP_by_cn(userData.get("name").toString().replace("
", "_")+"_"+ userData.get("lastname").toString().replace(" ", "_"),conexion_LDAP);
             //conecto a LDAP
             lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
             //autentico
             lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));
             //borro entrada
             lc.delete(dn);
             //desconecto
             lc.disconnect();

       } catch (LDAPException e) {
             errorList.put(getBundleString("popupLDAPDeleteUserError"));
             e.printStackTrace();
       } catch (UnsupportedEncodingException e) {
             errorList.put(getBundleString("popupErrorDB"));
             e.printStackTrace();
       }
       catch(IllegalArgumentException e ){
             e.printStackTrace();
             if(e.toString().contains("DN_PARAM_ERROR"))
             errorList.put(getBundleString("popupLDAPDeleteUserError"));
       }
       if (errorList.length() > 0) {
             throw new UsersServerBeanException(errorList.toString());
       }
}
// función para añadir nuevo usuario LDAP

/** newUserLDAP
  * @param userData: HashMap que contiene los datos de la entrada LDAP
  * @param conexión_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: result: String = Null si la entrada se creó correctamente
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPNewUserError: No se ha podido insertar el usuario en LDAP.
  */
```

```
public String newUserLDAP(HashMap userData, HashMap conexion_LDAP) throws
UsersServerBeanException {
       JSONArray errorList = new JSONArray();
       String result = null;

       try {

        LDAPConnection lc = new LDAPConnection();
        UsersServerBean userBean=new UsersServerBean();
        lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
             //autentico
        lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));

        UsersServerBean UserBean=new UsersServerBean();

        LDAPAttributeSet attributeSet = new LDAPAttributeSet();

              //insertando objectclass de LDAP y demás atributos de la entrada
            LDAPAttribute objC=new LDAPAttribute("objectclass");
            objC.addValue("inetOrgPerson");
            objC.addValue("asteriskSIPUser");
            attributeSet.add(objC);
            attributeSet.add(new LDAPAttribute("cn", userData.get("name").toString().replace("
", "_")+"_"+ userData.get("lastname").toString().replace(" ", "_")));
            attributeSet.add(new LDAPAttribute("givenname",userData.get("name").toString()));
            attributeSet.add(new LDAPAttribute("sn", userData.get("lastname").toString()));
            attributeSet.add(new LDAPAttribute("mail",userData.get("email").toString()));
            attributeSet.add(new
LDAPAttribute("userpassword",userData.get("password").toString()));
            attributeSet.add(new LDAPAttribute("uid",userData.get("name").toString()+"
"+userData.get("lastname").toString()));
            attributeSet.add(new
LDAPAttribute("preferredLanguage",userData.get("language").toString()));


        //parametros asterisk
        attributeSet.add(new
LDAPAttribute("AstAccountName",userData.get("name").toString().replace(" ", "_")+"_"+
userData.get("lastname").toString().replace(" ", "_")));
        attributeSet.add(new
LDAPAttribute("AstAccountCallerID",userData.get("userName").toString()));
        attributeSet.add(new
LDAPAttribute("AstAccountContext",userData.get("group").toString()));
        attributeSet.add(new
LDAPAttribute("AstAccountSecret",userData.get("password").toString()));

        ArrayList types=UserBean.getUserTypes("order by id_tipo");
        while(!types.isEmpty()){
             HashMap type=(HashMap)types.get(0);
             if(type.get("id_type").toString().equals(userData.get("id_type").toString())){
                  attributeSet.add(new
LDAPAttribute("AstAccountType",type.get("type").toString())); //user, peer, friend
                  break;
             }
             types.remove(0);
```

```
        }

        if(userData.get("canreinvite").toString().equals("true")){
              attributeSet.add(new LDAPAttribute("AstAccountCanReinvite","yes"));
        }
        else{
              attributeSet.add(new LDAPAttribute("AstAccountCanReinvite","no"));
        }
        if(userData.get("nat").toString().equals("true")){
              attributeSet.add(new LDAPAttribute("AstAccountNAT","yes"));
        }
        else{
              attributeSet.add(new LDAPAttribute("AstAccountNAT","no"));
        }
        if(userData.get("voicemail").toString().equals("true")){
             attributeSet.add(new
LDAPAttribute("AstAccountMailbox",userData.get("email").toString()));
        }

        LDAPAttribute astC=new LDAPAttribute("AstAccountAllowedCodec");
        CodecsServerBean CodecsBean=new CodecsServerBean();
        ArrayList codecs=CodecsBean.getCodecs("order by id_codec");
        boolean fAstAccountAllowedCodec=false;
        while(!codecs.isEmpty()){
             HashMap codec=(HashMap)codecs.get(0);
             if(userData.get("codecs").toString().contains(codec.get("id_codec").toString())){
                   astC.addValue(codec.get("codec").toString());//user, peer, friend
                   fAstAccountAllowedCodec=true;
             }
             codecs.remove(0);
        }
        if(fAstAccountAllowedCodec==true)
             attributeSet.add(astC);

        //atributos fijos
        attributeSet.add(new LDAPAttribute("AstAccountDisallowedCodec","all"));
        attributeSet.add(new LDAPAttribute("AstAccountHost","dynamic"));

        //construyo dn
        //String  dn  =
"uid="+userData.get("userName").toString()+",ou="+userData.get("profile").toString()+","+
containerName;
        String  dn  = "uid="+userData.get("name").toString()+"
"+userData.get("lastname").toString()+",ou=usuariosIRI,"+
"dc="+conexion_LDAP.get("ldapDomain").toString().substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length());

      //creo nueva entrada LDAP
             LDAPEntry newEntry = new LDAPEntry( dn, attributeSet );
             //añado entrada
             lc.add(newEntry);
             //desconecto
             lc.disconnect();

       } catch ( LDAPException e) {
             e.printStackTrace();
             errorList.put(getBundleString("popupLDAPNewUserError"));
       }
       catch( UnsupportedEncodingException e ){
             e.printStackTrace();
             errorList.put(getBundleString("popupErrorDB"));
```

```
	} catch (CodecsServerBeanException e) {//para LDAP
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	}

	if (errorList.length() > 0) {
		throw new UsersServerBeanException(errorList.toString());
	}

	return result;
}
// función para buscar un usuario dado su cn en LDAP. Devuelve su dn si lo encuentra
/** search_dn_UserLDAP_by_cn
  * @param cn: String que contiene el cn de la entrada LDAP
  * @param conexion_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: result: String = dn de la entrada buscada
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPConexionError: Conexión LDAP fallida.
  */

public String search_dn_UserLDAP_by_cn(String cn, HashMap conexion_LDAP) throws
UsersServerBeanException {
	JSONArray errorList = new JSONArray();
	String result=null;

	try {
	  LDAPConnection lc = new LDAPConnection();
	 UsersServerBean userBean=new UsersServerBean();
	  lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
		//autenticación
	  lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));
	  LDAPSearchResults searchResults
=lc.search("dc="+conexion_LDAP.get("ldapDomain").toString().substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()), LDAPConnection.SCOPE_SUB, "cn="+cn,
null, false);

	  LDAPAttributeSet attributeSet = new LDAPAttributeSet();

		//hacer búsqueda de posibles usuarios existentes en LDAP
	  while ( searchResults.hasMore()) {
		LDAPEntry nextEntry = null;
			nextEntry = searchResults.next();
			result=nextEntry.getDN();
	  }
	} catch ( LDAPException e) {
		e.printStackTrace();
		errorList.put(getBundleString("popupLDAPConexionError"));
	}
	catch( UnsupportedEncodingException e ){
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	}
	if (errorList.length() > 0) {
```

```
              throw new UsersServerBeanException(errorList.toString());
       }
       return result;
}
// busca la password de una entrada LDAP definida por su cn

/** search_pw_UserLDAP_by_cn
  * @param cn: String que contiene el cn de la entrada LDAP
  * @param conexion_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: result: String = password de la entrada buscada
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPConexionError: Conexión LDAP fallida.
  */

public String search_pw_UserLDAP_by_cn(String cn, HashMap conexion_LDAP) throws
UsersServerBeanException {
       JSONArray errorList = new JSONArray();
       String result=null;

       try {
        LDAPConnection lc = new LDAPConnection();
              UsersServerBean userBean=new UsersServerBean();
        lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
             //autenticación
        lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));
        LDAPSearchResults searchResults
=lc.search("dc="+conexion_LDAP.get("ldapDomain").toString().substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()), LDAPConnection.SCOPE_SUB, "cn="+cn,
null, false);

        LDAPAttributeSet attributeSet = new LDAPAttributeSet();

            //hacer búsqueda de posibles usuarios existentes en LDAP
        while ( searchResults.hasMore()) {
             LDAPEntry nextEntry = null;
                   nextEntry = searchResults.next();
                   result=nextEntry.getAttribute("userPassword").getStringValue();
        }
       } catch ( LDAPException e) {
             e.printStackTrace();
             errorList.put(getBundleString("popupLDAPConexionError"));
       }
       catch( UnsupportedEncodingException e ){
             e.printStackTrace();
             errorList.put(getBundleString("popupErrorDB"));
       }
       if (errorList.length() > 0) {
             throw new UsersServerBeanException(errorList.toString());
       }
       return result;
}

//
```

```
/** configLDAP
  * @param parametersLDAP: HashMap que contiene los parámetros de la conexión LDAP
  * @return: devuelve una conexión LDAP con los parámetros definidos en la entrada
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPConexionError: Conexión LDAP fallida.
  */
public void configLDAP(HashMap parametersLDAP) throws UsersServerBeanException {
       JSONArray errorList = new JSONArray();
       String result=null;

       try {
                     if (userEJB == null) {
                            userEJB = asModule.UsersManagerUtil.getHome().create();
                     }
                     result = userEJB.newConexionLDAP(parametersLDAP);

                     if (result == null) {
                            errorList.put(getBundleString("popupErrorDB"));
                            throw new UsersServerBeanException(errorList.toString());
                     }

              String ldapHost=parametersLDAP.get("ldapHost").toString();
              String ldapBase="dc="+parametersLDAP.get("ldapDomain").toString().substring(0,
parametersLDAP.get("ldapDomain").toString().indexOf("."))+",dc="+parametersLDAP.get("ldapDomai
n").toString().substring(parametersLDAP.get("ldapDomain").toString().indexOf(".")+1,
parametersLDAP.get("ldapDomain").toString().length());
              int ldapPort;
              if(parametersLDAP.get("ldapPort").toString().equals("DEFAULT_PORT")){
                     ldapPort=LDAPConnection.DEFAULT_PORT;
              }
              else{
                     ldapPort=Integer.parseInt(parametersLDAP.get("ldapPort").toString());
              }

              String cn=parametersLDAP.get("cn").toString();
              String password_cn=parametersLDAP.get("password_cn").toString();
              int ldapVersion  = LDAPConnection.LDAP_V3;

         LDAPConnection lc = new LDAPConnection();

         //búsqueda de posibles usuarios existentes en LDAP
         lc.connect( ldapHost, ldapPort );
         lc.bind(ldapVersion, "cn="+cn+","+ldapBase, password_cn.getBytes("UTF8"));
         LDAPAttributeSet attributeSet = new LDAPAttributeSet();

         //creo nueva entrada LDAP
         String  dn = "ou=usuariosIRI,"+ ldapBase;
              LDAPEntry newEntry = new LDAPEntry( dn, attributeSet );
         LDAPAttribute objC=new LDAPAttribute("objectclass");
             objC.addValue("organizationalUnit");
             attributeSet.add(objC);
             attributeSet.add(new LDAPAttribute("ou", "usuariosIRI"));
             lc.add(newEntry);

   //desconecto
          lc.disconnect();

       } catch ( LDAPException e) {
              e.printStackTrace();
              if(!e.toString().contains("La Entrada Existe")){ //la conexión ya fue creada
                     errorList.put(getBundleString("popupLDAPConexionError"));
              }
```

```
	}
	catch( UnsupportedEncodingException e ){
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	} catch (RemoteException e) {
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	} catch (CreateException e) {
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	} catch (NamingException e) {
		e.printStackTrace();
		errorList.put(getBundleString("popupErrorDB"));
	}

	if (errorList.length() > 0) {
		throw new UsersServerBeanException(errorList.toString());
	}
}
// función para modificar usuario LDAP con los nuevos parámetros de Asterisk

/** modifyUserLDAP
  * @param dn_search: HashMap que contiene los datos del cn de la entrada LDAP
  * @param userData: HashMap que contiene los datos del cn de la entrada LDAP
  * @param conexion_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: result: String = Null si la operación fue correcta
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPModifyUserError: No se ha podido modificar el usuario en LDAP
  */

public String modifyUserLDAP(String dn_search,HashMap userData, HashMap conexion_LDAP) throws
UsersServerBeanException {
	JSONArray errorList = new JSONArray();
	String result = null;

	try {
	  LDAPConnection lc = new LDAPConnection();
	  UsersServerBean userBean=new UsersServerBean();

	  // búsqueda de posibles usuarios existentes en LDAP
	  lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
		//autentico
	  lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));

	  ArrayList modList = new ArrayList();
	  LDAPModification modifyEntry = new LDAPModification();
	  LDAPAttribute attribute;
		//parametros inetorgPerson
	  attribute = new LDAPAttribute("cn", userData.get("name").toString().replace(" ",
"_")+"_"+ userData.get("lastname").toString().replace(" ", "_"));
	  modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
	  attribute = new LDAPAttribute("givenname",userData.get("name").toString());
	  modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
	  attribute = new LDAPAttribute("sn", userData.get("lastname").toString());
	  modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
```

```
        attribute = new LDAPAttribute("mail",userData.get("email").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("userpassword",userData.get("password").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new
LDAPAttribute("preferredLanguage",userData.get("language").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));


        //parametros asterisk
        attribute = new
LDAPAttribute("AstAccountName",userData.get("name").toString().replace(" ", "_")+"_"+
userData.get("lastname").toString().replace(" ", "_"));
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new
LDAPAttribute("AstAccountCallerID",userData.get("userName").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountContext",userData.get("group").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountSecret",userData.get("password").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        userBean=new UsersServerBean();
        ArrayList types=userBean.getUserTypes("order by id_tipo");
        while(!types.isEmpty()){
             HashMap type=(HashMap)types.get(0);
             if(type.get("id_type").toString().equals(userData.get("id_type").toString())){
                  attribute = new
LDAPAttribute("AstAccountType",type.get("type").toString());
               modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
                  break;
             }
             types.remove(0);
        }

        if(userData.get("canreinvite").toString().equals("true")){
              attribute = new LDAPAttribute("AstAccountCanReinvite","yes");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        }
        else{
              attribute = new LDAPAttribute("AstAccountCanReinvite","no");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        }
        if(userData.get("nat").toString().equals("true")){
              attribute = new LDAPAttribute("AstAccountNAT","yes");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        }
        else{
              attribute = new LDAPAttribute("AstAccountNAT","no");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        }

        attribute = new LDAPAttribute("AstAccountMailbox",userData.get("email").toString());
          modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        if(!userData.get("voicemail").toString().equals("true")){
             attribute = new LDAPAttribute("AstAccountMailbox");
          modList.add( new LDAPModification(LDAPModification.DELETE, attribute));
        }

        LDAPAttribute astC=new LDAPAttribute("AstAccountAllowedCodec");
        LDAPAttribute astC_user=new LDAPAttribute("AstAccountAllowedCodec");
```

```
        CodecsServerBean CodecsBean=new CodecsServerBean();
        ArrayList codecs=CodecsBean.getCodecs("order by id_codec");
        boolean fAstAccountAllowedCodec=false;
        while(!codecs.isEmpty()){
             HashMap codec=(HashMap)codecs.get(0);
             astC.addValue(codec.get("codec").toString());

if(!userData.get("codecs").toString().contains(codec.get("id_codec").toString())){
                   astC_user.addValue(codec.get("codec").toString());
                   fAstAccountAllowedCodec=true;
             }
             codecs.remove(0);
        }
        attribute=astC;
             modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
             if(fAstAccountAllowedCodec==true){
             attribute=astC_user;
             modList.add( new LDAPModification(LDAPModification.DELETE, attribute));
        }

        //fijos
        attribute = new LDAPAttribute("AstAccountDisallowedCodec","all");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountHost","dynamic");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        //creo nueva entrada LDAP
        LDAPModification[] mods = new LDAPModification[modList.size()];
        mods = (LDAPModification[])modList.toArray(mods);

             //añado entrada
             lc.modify(dn_search, mods);
             //desconecto
             lc.disconnect();

      } catch ( LDAPException e) {
             e.printStackTrace();
             errorList.put(getBundleString("popupLDAPModifyUserError"));
      }
      catch(IllegalArgumentException e ){
             e.printStackTrace();
             if(e.toString().contains("DN_PARAM_ERROR"))
             errorList.put(getBundleString("popupLDAPModifyUserError"));
      }
      catch( UnsupportedEncodingException e ){
             e.printStackTrace();
             errorList.put(getBundleString("popupErrorDB"));
      } catch (CodecsServerBeanException e) {
             e.printStackTrace();
             errorList.put(getBundleString("popupErrorDB"));
      }

      if (errorList.length() > 0) {
             throw new UsersServerBeanException(errorList.toString());
      }

      return result;
}

// comprobar si existe un usuario en LDAP
/** existUserLDAP
  * @param dn_search: HashMap que contiene los datos del cn de la entrada LDAP
  * @param userData: HashMap que contiene los datos del cn de la entrada LDAP
```

```
  * @param conexion_LDAP: HashMap que contiene los datos de la conexión LDAP
  * @return: result: String = Null si la operación fue correcta
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  * @throws: popupLDAPModifyUserError: No se ha podido modificar el usuario en LDAP
  */

public String existUserLDAP(String dn_search,HashMap userData,HashMap conexion_LDAP) throws
UsersServerBeanException {
       JSONArray errorList = new JSONArray();
       String result = null;

       try {
         LDAPConnection lc = new LDAPConnection();
         UsersServerBean userBean=new UsersServerBean();
         lc.connect(conexion_LDAP.get("ldapHost").toString(),
Integer.parseInt(conexion_LDAP.get("ldapPort").toString()));
              //autentico
         lc.bind(LDAPConnection.LDAP_V3,
"cn="+conexion_LDAP.get("cn").toString()+","+"dc="+conexion_LDAP.get("ldapDomain").toString().
substring(0,
conexion_LDAP.get("ldapDomain").toString().indexOf("."))+",dc="+conexion_LDAP.get("ldapDomain"
).toString().substring(conexion_LDAP.get("ldapDomain").toString().indexOf(".")+1,
conexion_LDAP.get("ldapDomain").toString().length()),
conexion_LDAP.get("password_cn").toString().getBytes("UTF8"));
        ArrayList modList = new ArrayList();
        LDAPModification modifyEntry = new LDAPModification();

        LDAPAttribute attribute;
        LDAPAttribute objC=new LDAPAttribute("objectclass");
            objC.addValue("asteriskSIPUser");
            modList.add( new LDAPModification(LDAPModification.ADD, objC));

             //cambio cn para quitar espacios
            attribute = new LDAPAttribute("cn",userData.get("name").toString().replace(" ",
"_")+"_"+ userData.get("lastname").toString().replace(" ", "_"));
            modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
            //añado atributos asterisk
            attribute = new
LDAPAttribute("AstAccountName",userData.get("name").toString().replace(" ", "_")+"_"+
userData.get("lastname").toString().replace(" ", "_"));
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new
LDAPAttribute("AstAccountCallerID",userData.get("userName").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountContext",userData.get("group").toString());
        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountSecret",userData.get("password").toString());

        modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        ArrayList types=userBean.getUserTypes("order by id_tipo");
        while(!types.isEmpty()){
             HashMap type=(HashMap)types.get(0);
             if(type.get("id_type").toString().equals(userData.get("id_type").toString())){
              attribute = new LDAPAttribute("AstAccountType",type.get("type").toString());
               modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
                break;
             }
             types.remove(0);
        }

        if(userData.get("canreinvite").toString().equals("true")){
```

```
                attribute = new LDAPAttribute("AstAccountCanReinvite","yes");
            modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        }
        else{
                attribute = new LDAPAttribute("AstAccountCanReinvite","no");
            modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        }
        if(userData.get("nat").toString().equals("true")){
                attribute = new LDAPAttribute("AstAccountNAT","yes");
            modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        }
        else{
                attribute = new LDAPAttribute("AstAccountNAT","no");
            modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        }

        attribute = new LDAPAttribute("AstAccountMailbox",userData.get("email").toString());
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        if(!userData.get("voicemail").toString().equals("true")){
             attribute = new LDAPAttribute("AstAccountMailbox");
           modList.add( new LDAPModification(LDAPModification.DELETE, attribute));
        }

        LDAPAttribute astC=new LDAPAttribute("AstAccountAllowedCodec");
        LDAPAttribute astC_user=new LDAPAttribute("AstAccountAllowedCodec");
        CodecsServerBean CodecsBean=new CodecsServerBean();
        ArrayList codecs=CodecsBean.getCodecs("order by id_codec");
        boolean fAstAccountAllowedCodec=false;
        while(!codecs.isEmpty()){
             HashMap codec=(HashMap)codecs.get(0);
             astC.addValue(codec.get("codec").toString());

if(!userData.get("codecs").toString().contains(codec.get("id_codec").toString())){
                    astC_user.addValue(codec.get("codec").toString());
                    fAstAccountAllowedCodec=true;
             }
             codecs.remove(0);
        }
        attribute=astC;
             modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
             if(fAstAccountAllowedCodec==true){
             attribute=astC_user;
             modList.add( new LDAPModification(LDAPModification.DELETE, attribute));
        }

        //fijos
        attribute = new LDAPAttribute("AstAccountDisallowedCodec","all");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
        attribute = new LDAPAttribute("AstAccountHost","dynamic");
           modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));
           //aÑade grupo iri
           attribute = new LDAPAttribute( new LDAPAttribute("ou","usuariosIRI"));
              modList.add( new LDAPModification(LDAPModification.REPLACE, attribute));

        //creo nueva entrada LDAP
        LDAPModification[] mods = new LDAPModification[modList.size()];
        mods = (LDAPModification[])modList.toArray(mods);

             //añado entrada
             lc.modify(dn_search, mods);
             //desconecto
```

```
                lc.disconnect();


        } catch ( LDAPException e) {
                e.printStackTrace();
                errorList.put(getBundleString("popupLDAPModifyUserError"));
        }
        catch( UnsupportedEncodingException e ){
                e.printStackTrace();
                errorList.put(getBundleString("popupErrorDB"));
        } catch (CodecsServerBeanException e) {
                e.printStackTrace();
                errorList.put(getBundleString("popupErrorDB"));
        }

        if (errorList.length() > 0) {
                throw new UsersServerBeanException(errorList.toString());
        }

        return result;
}

/** getConexionLDAP
  * @return: result: Hashmap = relleno con los parámetros de la conexión LDAP
  * @throws: popupErrorDB: Ha ocurrido un error inesperado de comunicación con el
servidor
  */

public HashMap getConexionLDAP() throws UsersServerBeanException {
        JSONArray errorList = new JSONArray();
        HashMap result = new HashMap();

        try {
                if (userEJB == null) {
                        userEJB = asModule.UsersManagerUtil.getHome().create();
                }
                result = userEJB.getConexionLDAP();

                if (result == null) {
                        errorList.put(getBundleString("popupErrorDB"));
                }
        } catch (RemoteException e) {
                e.printStackTrace();
                errorList.put(getBundleString("popupErrorDB"));
        } catch (CreateException e) {
                e.printStackTrace();
                errorList.put(getBundleString("popupErrorDB"));
        } catch (NamingException e) {
                e.printStackTrace();
                errorList.put(getBundleString("popupErrorDB"));
        }

        if (errorList.length() > 0) {
                throw new UsersServerBeanException(errorList.toString());
        }
        return result;
}
}
//*************FIN METODOS LDAP********************/
```

Messages_es_ES.properties:

Mensajes lanzados en las excepciones:

```
########################################         GENERALES
##############################
…


########################################         SERVER ERRORS
##############################
…
popupErrorDB= Ha ocurrido un error inesperado de comunicación con el servidor. Pongase en contacto con su administrador.
…

########################################         POPUPS MESSAGES
##############################
…
popupLDAPMessageSuccess=Conexión LDAP correcta.
popupLDAPConexionError=Conexión LDAP fallida.
popupLDAPNewUserError=No se ha podido insertar el usuario en LDAP.
popupLDAPModifyUserError=No se ha podido modificar el usuario en LDAP
popupLDAPDeleteUserError=No se ha podido eliminar el usuario en LDAP

########################################        LEFT MENU         ##############################
…
menuAdminLDAP=LDAP

…

menuAdminEditLDAP=Configuración LDAP

…

########################################      LDAPconf.JSP
##############################
editLDAPConfPageTitle=IRI-421 "Configuración LDAP"
LDAPConfContentTitle=Configuración LDAP
LDAPconfTableTitle=Datos cuenta LDAP
editHostLDAP=Dirección LDAP
editBaseLDAP=Dominio LDAP
editPortLDAP=Puerto LDAP
editDNAdminLDAP=CN Administrador
editPasswordDN=Contraseña
```

editUsers.js:

Este código se ejecuta en las vistas de administración de usuarios, por lo tanto en él se utilizan todas las funciones defininas anteriormente para realizar la autenticación contra LDAP en las operaciones de creación, búsqueda, modificación y borrado de usuarios.

Los datos de un usuario se guardan en una variable de tipo ***Hashmap***, con los siguientes campos:

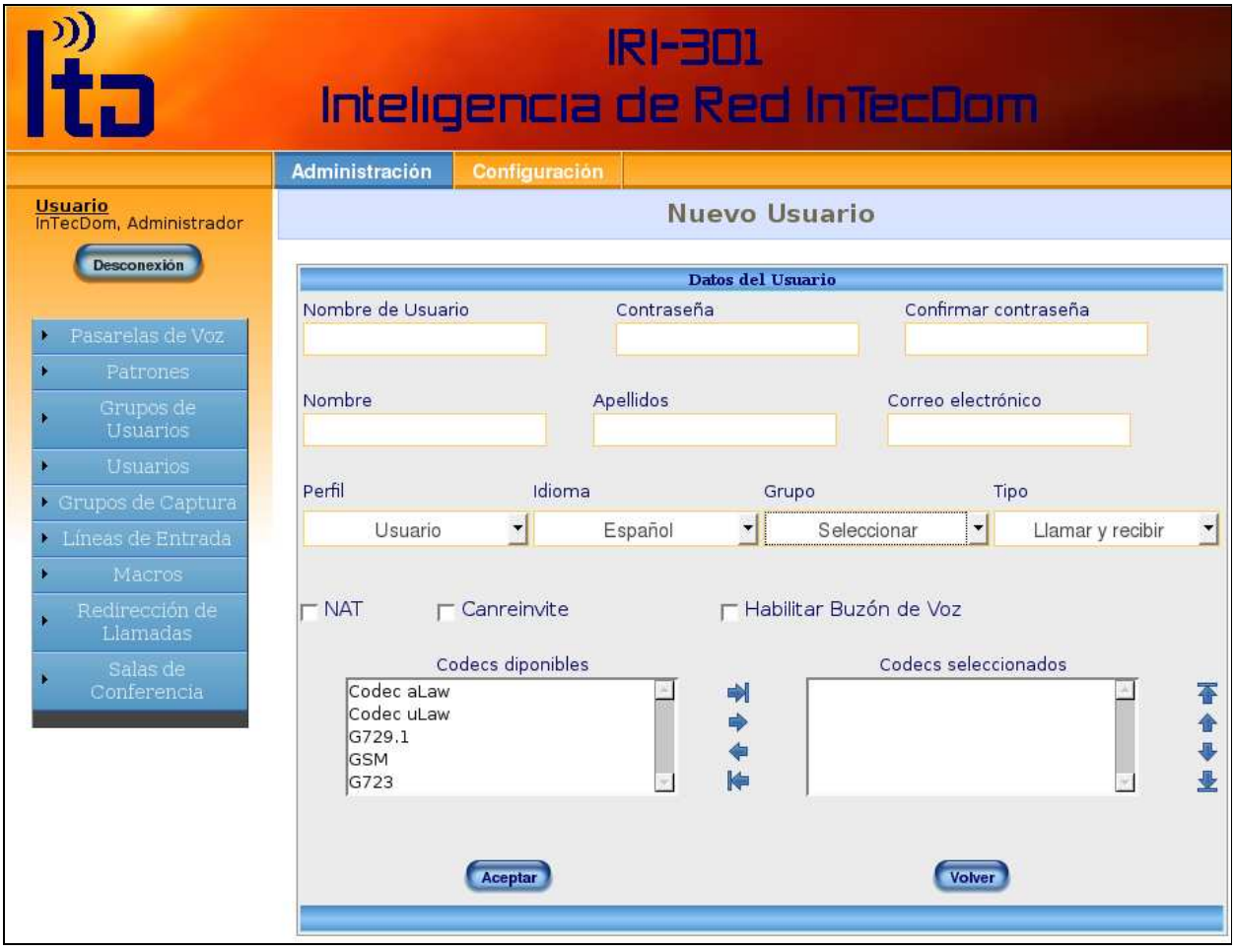
Campo	Tipo	Descripción
userName	String	identificador de usuario
id_profile	Integer	Perfil de usuario: 1 = administrador (acceso total) 2 = usuario (acceso de usuario)
id_type	Integer	Perfil de usuario: 1 = friend (llamar y recibir) 2 = peer (sólo llamar) 3 = user (sólo recibir)
id_language	Integer	Idioma de usuario: 1 = Español 2 = Inglés
id_group	Integer	Grupo de usuario: 1 = grupo_test …
Name	String	Nombre
lastname	String	Apellidos
Password	String	Contraseña
confirmPassword	String	Contraseña
Email	String	Correo electrónico
Canreinvite	String	Opción de telefonía canreinvite
Nat	String	Opción de telefonía nat
Voicemail	String	Buzón de voz
codecs	ArrayList de String	Codecs: 1 = alaw 2 = ulaw 3 = g729

En el sistema de telefonía existente, los datos de los nuevos usuarios son rellenados en la página Web y desde ahí son leídos por diversas funciones y guardados en los campos de la tabla hash de la variable usuario.

Este mismo funcionamiento se ha aplicado para LDAP. Los datos almacenados en la tabla *hash* de la variable ***usuario*** son utilizados para realizar la conexión y autenticación con el directorio.

Como vemos en la tabla *hash* descrita anteriormente, hay varios campos cuyo valor está limitado a ciertas opciones que están numeradas y almacenadas en distintas tablas de la base de datos PostgreSQL. A la hora de desarrollar el código para LDAP se prefirió guardar estas variables como texto en lugar de como enteros. De este modo la información almacenada en LDAP será más descriptiva.

A continuación se muestra el extracto del código de las funciones *getUserInfo, getUserCodecs, isUserCodec, getProfiles, getLanguages, getGroups, getUserTypes, getUserCodecs* definidas en el fichero **editUsers.js**, en la que se recoge la información de usuario que ha sido introducida por medio de la interfaz Web en la página mostrada a continuación y guardada en la tabla hashmap anterior, y a continuación las funciones *onLoad* y *ApplyChanges* del fichero **LDAPconf.js**. Ambos ficheros pertenecen respectivamente a *Users* y *SystemConfig*, que a su vez forman parte de los java *servlets* contenidos en el fichero *WebApp* correspondiente al contenedor *IRI-Web*.



editUsers.js:

```
//definición de variables

var id_user = null;
var id_group = null;
var user =
{javaClass:'java.util.HashMap',map:{userName:'',id_profile:'',id_type:'',id_language:'',id_gro
up:'', name:'',lastname:'',password:'',confirmPassword:'', email:'', canreinvite:'', nat:'',
voicemail:'', codecs:{javaClass:'java.util.ArrayList',list:[]}}};
var userToSend =
{javaClass:'java.util.HashMap',map:{userName:'',id_profile:'',id_type:'',id_language:'',id_gro
up:'', name:'',lastname:'',password:'',confirmPassword:'', email:'', canreinvite:'', nat:'',
voicemail:'', keepExtensions:'', codecs:{javaClass:'java.util.ArrayList',list:[]}}};
var jsonrpc = null;
var userCodecs = null;
…

//definición de funciones
/*****************************************************************************/
…
/** getUserInfo
  * @param id_user: id del usuario creado por el interfaz Web del que queremos los datos
  * @return: result: user
  * @throws: showGenericError
  */
function getUserInfo(id_user){
var getUserInfo_response =  function(result, exception){
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }
        user = result;
        $('userNameText').value = user.map.userName;
        $('nameText').value = user.map.name;
        $('lastnameText').value = user.map.lastname;
        $('emailText').value = user.map.email;
        $('userPasswordText').value = '';
        $('confirmUserPasswordText').value = '';
        if(user.map.nat == true){
             $('natCheck').checked = true;
        }else{
             $('natCheck').checked = false;
        }
        if(user.map.canreinvite == true){
             $('canreinviteCheck').checked = true;
        }else{
             $('canreinviteCheck').checked = false;
        }
        if(user.map.voicemail == true){
             $('voiceMailCheck').checked = true;
        }else{
             $('voiceMailCheck').checked = false;
        }
        id_group = user.map.id_group;

        if(alreadyLoaded()) hideProgress();

};
```

```
jsonrpc.usersBean.getUserInfo(getUserInfo_response, id_user);
}

/** getUserCodecs
  * @param id_user: id del usuario creado por el interfaz Web del que queremos los datos
  * @return: result: userCodecs
  * @throws: showGenericError
  */
function getUserCodecs(id_user){
var getUserCodecs_response =  function(result, exception){
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }
        userCodecs = result;

        if(alreadyLoaded()) hideProgress();
};

jsonrpc.usersBean.getUserCodecs(getUserCodecs_response, id_user);
}

/** isUserCodec
  * @param id_codec: id del codec elegido por el interfaz Web
  * @return: result: found. Indica si el codec elegido es uno de los codecs de la lista
  */
function isUserCodec(id_codec){
        var found = false;

        for(var i = 0; i < userCodecs.list.length && !found; i++){
                found = (userCodecs.list[i].map.id_codec == id_codec);
        }

        return found;
}
/*****************************************************************************/
/** getProfiles
  * @return: result: codecs
  * @throws: showGenericError
  */
function getProfiles(){
var getProfiles_response=function(result, exception) {
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }

    var profiles = result;
        initProfiles(profiles);

};

var initProfiles = function(profiles){
        var selectedList=$('selectProfile');

        while(selectedList.length > 0){
                selectedList.options[0] = null;
        }

        var selectedIndex = 0;
        for(var i = 0; i < profiles.list.length; i++){
```

```
                        selectedList.options[selectedIndex] = new
Option(profiles.list[i].map.name, profiles.list[i].map.id_profile);
                        selectedIndex++;
        }

        if(id_user == null)
        {
                if(selectedIndex > 0){
                        selectedList.options[0].selected = true;
                }
        }else
                {
                        selectedList.value =user.map.id_profile;
                }

        if(alreadyLoaded()) hideProgress();
};

        jsonrpc.usersBean.getProfiles(getProfiles_response);
}
/*****************************************************************************/

/** getLanguages
  * @return: result: languages
  * @throws: showGenericError
  */
function getLanguages(){
var getLanguages_response = function(result, exception) {
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }

    var languages = result;
        initLanguages(languages);

};

var initLanguages = function(languages){
        var selectedList=$('selectLanguage');

        while(selectedList.length > 0){
                selectedList.options[0] = null;
        }

        var selectedIndex = 0;
        for(var i = 0; i < languages.list.length; i++){
                        selectedList.options[selectedIndex] = new
Option(languages.list[i].map.name, languages.list[i].map.id_language);
                        selectedIndex++;
        }

        if(id_user == null)
        {
                if(selectedIndex > 0){
                        selectedList.options[0].selected = true;
                }
        }else
                {
                        selectedList.value =user.map.id_language;
                }
```

```
        if(alreadyLoaded()) hideProgress();

};

        jsonrpc.usersBean.getLanguages(getLanguages_response);
}

/*****************************************************************************/
/** getGroups
  * @return: result: groups
  * @throws: showGenericError
  */
function getGroups(){
var getGroups_response = function(result, exception) {
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }
       var groups = result;
        initGroups(groups);
};
       var initGroups =  function(groups){
        var selectedList=$('selectGroup');

        while(selectedList.length > 0){
                selectedList.options[0] = null;
        }
       selectedList.options[0] = new Option(getBundleString('editUserSelect'), -1);

        var selectedIndex = 1;
        for(var i = 0; i < groups.list.length; i++){
                    if(!groups.list[i].map.groupIsHidden){
                selectedList.options[selectedIndex] = new Option(groups.list[i].map.groupName,
groups.list[i].map.id_group);
                selectedIndex++;
                    }
        }

        if(id_user == null)
        {
                if(selectedIndex > 0){
                        selectedList.options[0].selected = true;
                }
        }else
                {
                        selectedList.value =user.map.id_group;
                }

        if(alreadyLoaded()) hideProgress();
};

        jsonrpc.usersBean.getGroups(getGroups_response, "order by id_grupo");
}

/*****************************************************************************/
/** getUserTypes
  * @return: result: user Type
  * @throws: showGenericError
  */
function getUserTypes(){
var getUserTypes_response = function(result, exception) {
        if(exception){
                hideProgress();
```

```
                showGenericError();
                return;
        }

       var types = result;
        initUserTypes(types);
};

var initUserTypes =  function(types){
        var selectedList=$('selectUserType');

        while(selectedList.length > 0){
                selectedList.options[0] = null;
        }

selectedList.options[0] = new Option(getBundleString('editUserSelect'), -1);

        var selectedIndex = 1;
        for(var i = 0; i < types.list.length; i++){
                        selectedList.options[selectedIndex] = new
Option(types.list[i].map.typeDescription, types.list[i].map.id_type);
                        selectedIndex++;
        }

        if(id_user == null)
        {
                if(selectedIndex > 0){
                        selectedList.options[0].selected = true;
                }
        }else
                {
                        selectedList.value =user.map.id_type;
                }
       if(alreadyLoaded()) hideProgress();
};

        jsonrpc.usersBean.getUserTypes(getUserTypes_response, "order by id_tipo");
}

/*****************************************************************************/
/** getCodecs
  * @return: result: user Codecs
  * @throws: showGenericError
  */
function getCodecs(){
var getCodecs_response =  function(result, exception) {
        if(exception){
                hideProgress();
                showGenericError();
                return;
        }

       var codecs = result;
        initCodecs(codecs);

};
       var initCodecs =  function(codecs){
        var selectedList=$('selectedCodecs');
        var allList=$('allCodecs');
        while(selectedList.length > 0){
                selectedList.options[0] = null;
        }
```

```
        while(allList.length > 0){
                allList.options[0] = null;
        }

        var selectedIndex = 0;
        var allIndex = 0;
        if(id_user!=null){
        for(var i = 0; i < codecs.list.length; i++){
             if(!isUserCodec(codecs.list[i].map.id_codec)){
                  allList.options[allIndex] = new
Option(codecs.list[i].map.codecDescription, codecs.list[i].map.id_codec);
                allIndex++;
             }
        }

        for(var i = 0; i < userCodecs.list.length; i++){
                  selectedList.options[selectedIndex] = new
Option(userCodecs.list[i].map.codecDescription, userCodecs.list[i].map.id_codec);
                selectedIndex++;
        }
        }else{
             for(var i = 0; i < codecs.list.length; i++){
                  allList.options[allIndex] = new
Option(codecs.list[i].map.codecDescription, codecs.list[i].map.id_codec);
                allIndex++;
        }

        }
        if(alreadyLoaded()) hideProgress();
};

        jsonrpc.usersBean.getCodecs(getCodecs_response, "order by id_codec");
}

...

userToSend.map.userName = $('userNameText').value;
userToSend.map.id_profile =
$('selectProfile').options[$('selectProfile').selectedIndex].value;
//para LDAP
userToSend.map.profile = $('selectProfile').options[$('selectProfile').selectedIndex].text;
userToSend.map.id_language =
$('selectLanguage').options[$('selectLanguage').selectedIndex].value;
userToSend.map.language = $('selectLanguage').options[$('selectLanguage').selectedIndex].text;
userToSend.map.id_group = $('selectGroup').options[$('selectGroup').selectedIndex].value;
userToSend.map.group = $('selectGroup').options[$('selectGroup').selectedIndex].text;
//********
userToSend.map.name = $('nameText').value;
userToSend.map.lastname = $('lastnameText').value;
userToSend.map.password = $('userPasswordText').value;
userToSend.map.confirmPassword = $('confirmUserPasswordText').value;
userToSend.map.email = $('emailText').value;
userToSend.map.id_type = $('selectUserType').options[$('selectUserType').selectedIndex].value;
if($('natCheck').checked === true){
      userToSend.map.nat = "true";
}else{
      userToSend.map.nat = "false";
}
if($('canreinviteCheck').checked === true){
      userToSend.map.canreinvite = "true";
}else{
      userToSend.map.canreinvite = "false";
}
```

```
if($('voiceMailCheck').checked === true){
	userToSend.map.voicemail = "true";
}else{
	userToSend.map.voicemail = "false";
}

var selectedList=$('selectedCodecs');

for(var i = 0; i < selectedList.options.length; i++){
	userToSend.map.codecs.list[i] = Number(selectedList.options[i].value);

}

if(id_group !== null && id_group !=
$('selectGroup').options[$('selectGroup').selectedIndex].value){
	showChoice(getBundleString('popupKeepExtensionsTitle'),
getBundleString('popupKeepExtensionsMessage'),getBundleString('popupButtonYes'), keepExtens,
getBundleString('popupButtonNo'), dontKeepExtens);
}else{
	keepExtens();
}
}

/*****************************************************************************/
function volver(){
	 var  f = document.createElement("FORM");
	 var h = document.createElement("INPUT");

	 h.type='hidden';
	 h.name='id_user';
	 h.value=id_user;
	 f.id='formVolver';
	 f.method='post';
	 f.action='/iri/Users/listUsers.jsf';
	 f.appendChild(h);
	 document.body.appendChild(f);
	 $('formVolver').submit();
}

…
```

LDAPconf.js:

```
var id_user = null;
var id_group = null;
var parametersLDAP =
{javaClass:'java.util.HashMap',map:{ldapHost:'',ldapPort:'',ldapDomain:'',cn:'',password_cn:''}};
var jsonrpc = null;
var displayFlag = 0; //para comprobar si se cargo el contenido y esconder el gif del progreso
var counterFlag = 0; //para comprobar si se cargo el contenido y esconder el gif del progreso

/*****************************************************************************/
/** onLoad
  * @return: if successful continue, if error throws exception
  * @throws: showGenericError
  */
function onLoad(){
          onProgress(getBundleString('progressMessageLoad'));
          $('menuAdminAdministration').setAttribute("class", "item_active");
        id_user = isNaN(Number($F('id_user'))) ? null : Number($F('id_user'));
        jsonrpc = new JSONRpcClient("/iri/JSON-RPC");

          var getConfigLdap_response =  function(result, exception){
               if(exception){
                       hideProgress();
                       showGenericError();
                       return;
               }
               parametersLDAP = result;
               $('ldapHost').value=parametersLDAP.map.ldapHost;
               if(parametersLDAP.map.ldapPort=="389"){
                $('ldapPort').value="DEFAULT_PORT";
               }
               else{
                $('ldapPort').value=parametersLDAP.map.ldapPort;
               }
               $('ldapDomain').value=parametersLDAP.map.ldapDomain;
               $('cn').value=parametersLDAP.map.cn;

               //Las funciones alreadyLoaded() y hideProgress() están definidas en otros .js
               if(alreadyLoaded()) hideProgress();
          };
          jsonrpc.usersBean.getConexionLDAP(getConfigLdap_response);

}
/*****************************************************************************/
/** applyChanges
  * @return: if successful continue, if error throws exception
  * @throws: showGenericError
  */
function applyChanges(){
    onProgress(getBundleString('progressMessageApply'));
    var configLDAP_response =  function(result, exception){
          if(exception){
                if(exception.name == 'iri.serverBeans.UsersServerBean$UsersServerBeanException'){
                      hideProgress();
                      errorList = exception.message.parseJSON();
                      showErrors(getBundleString('popupErrorDBTitle'),errorList,
getBundleString('popupButtonAccept'), '');
                } else {
                      hideProgress();
                      showGenericError();
```

```
                }
                return;
        }
        if(alreadyLoaded()) hideProgress();

    showAlert(getBundleString('popupTitleSuccess'),getBundleString('popupLDAPMessageSuccess'),getB
undleString('popupButtonAccept'), back);
    };

    parametersLDAP.map.ldapHost = $('ldapHost').value;
    parametersLDAP.map.ldapDomain = $('ldapDomain').value;
    parametersLDAP.map.ldapPort = $('ldapPort').value;
    parametersLDAP.map.cn = $('cn').value;
    parametersLDAP.map.password_cn = $('password_cn').value;

    jsonrpc.usersBean.configLDAP(configLDAP_response,parametersLDAP);

    }
…
```

Por último detallamos las modificaciones efectuades en el fichero pom.xml, que se utiliza para identificar la información de proyecto Maven.

En este fichero la modificación efectuada por el desarrollo de este proyecto consiste en añadir una dependencia a la librería Novell de LDAP, y añadir el “artefacto” Maven jldap de versión 4.3

pom.xml:

```
…
- <!--  para novell openldap  -->
- <dependency>
  <groupId>com.novell.ldap</groupId>
  <artifactId>jldap</artifactId>
  <version>4.3</version>
  </dependency>
…
```

2. Manual de instalación de las herramientas del desarrollador

2.1. Manual de instalación de Eclipse y Maven

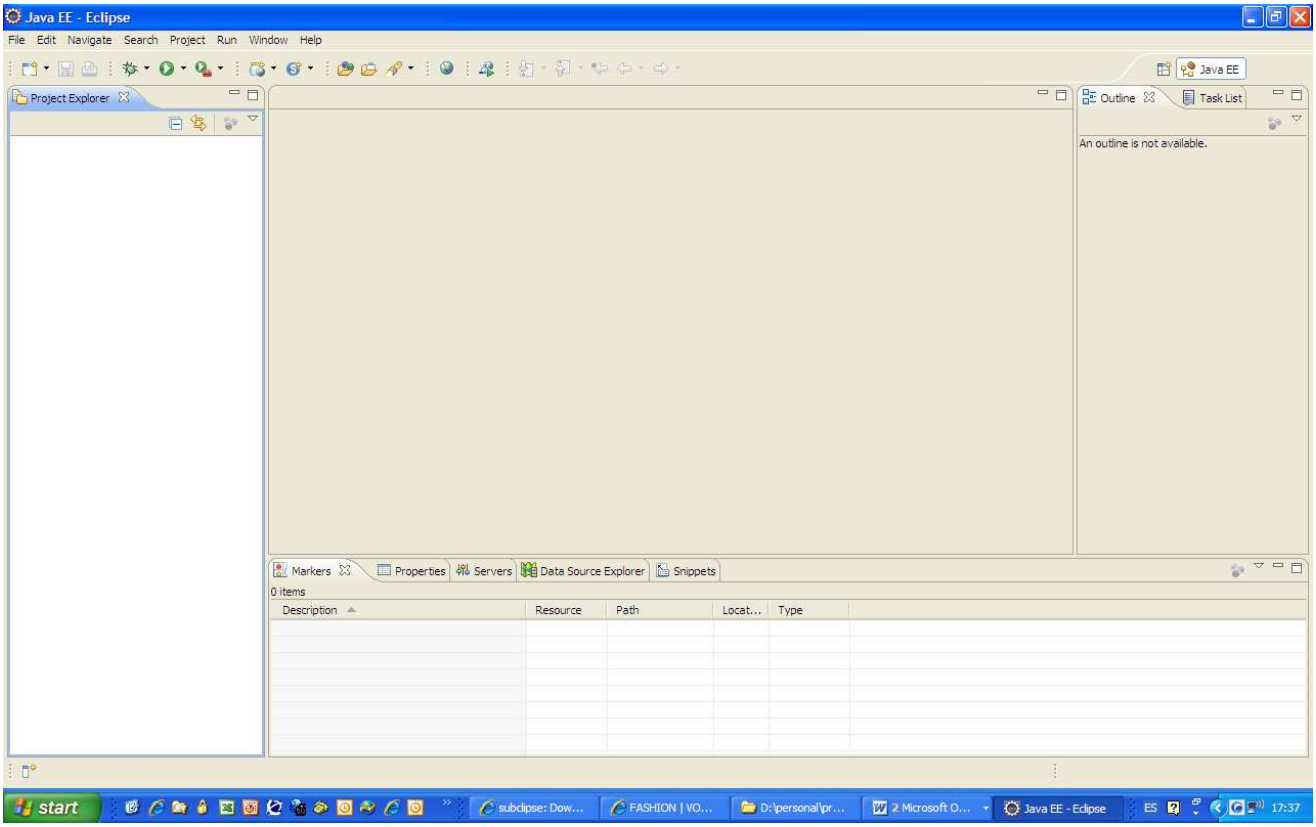
Para la realización del proyecto es necesario instalar en entorno de programación Eclipse.

En este caso Eclipse ha sido instalado en un ordenador con sistema operativo Windows XP.

Para instalar Eclipse en Windows, hay que descargar la última versión del correspondiente paquete de instalación de Eclipse para Windows, en este caso es Eclipse Galileo (disponible desde el 24 de Junio de 2009), desde la página Web oficial del proyecto:

http://www.eclipse.org/

Una vez finalizada la instalación, al ejecutar Eclipse obtendremos el siguiente entorno de programación para Java:



Tras la instalación, hay que instalar unos paquetes adicionales:

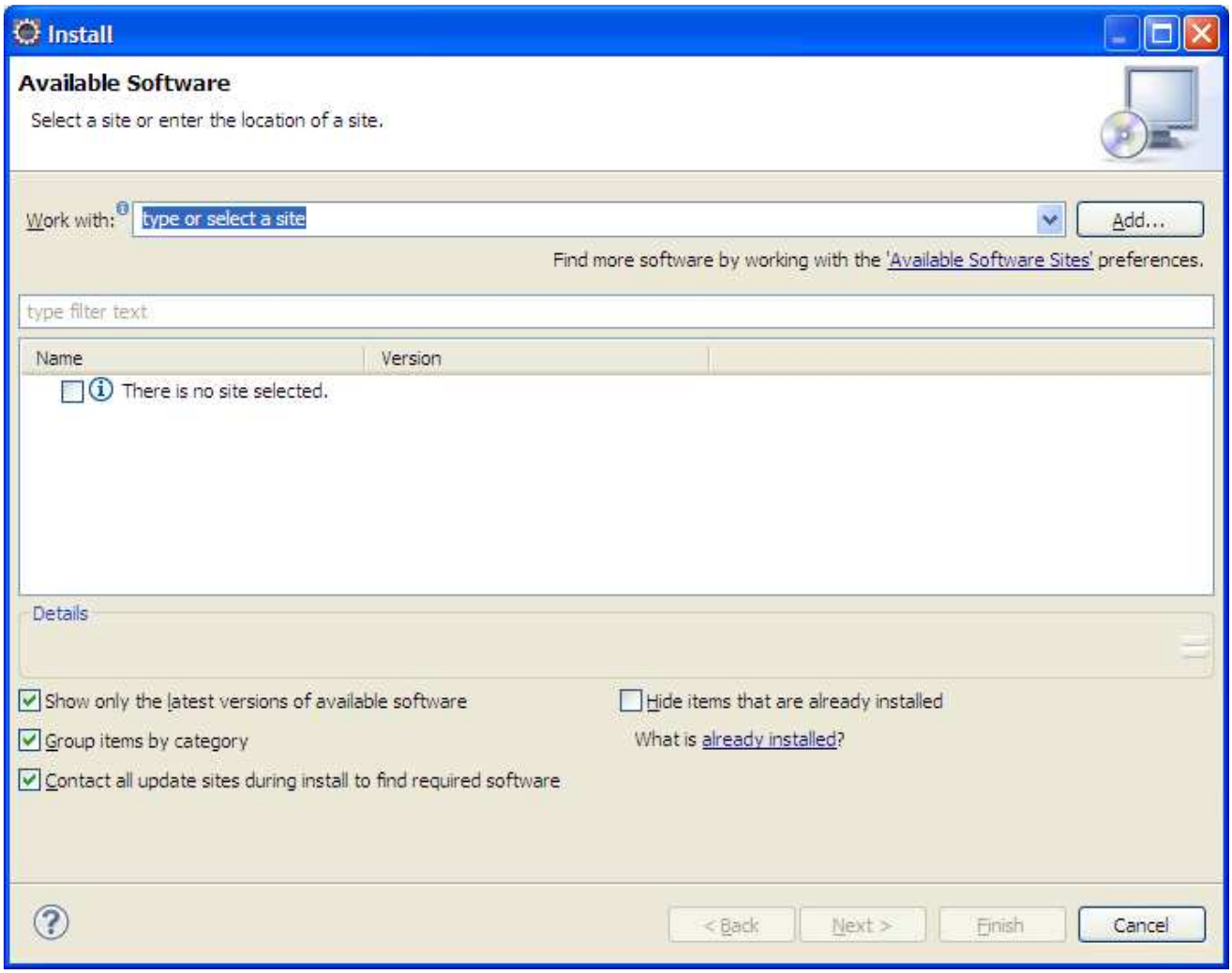
- subclipse

- Maven2

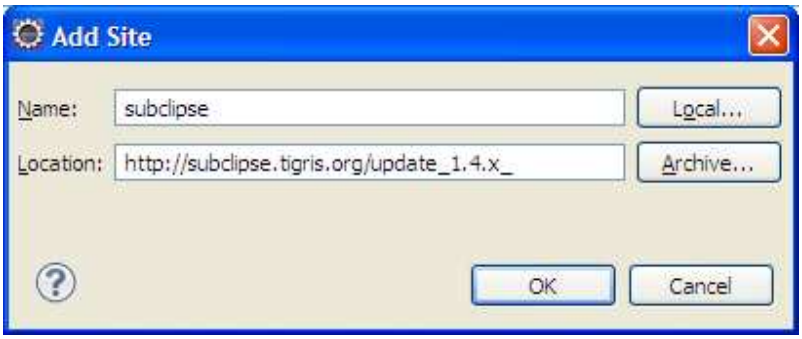
Estos paquetes se descargan desde las siguientes URLs:

- subclipse
  http://subclipse.tigris.org/update_1.4.x
- Maven2
  http://m2eclipse.sonatype.org/update/

Para instalar estos paquetes en Eclipse, una vez iniciada la aplicación Eclipse, hacer clic en *Help -> Install New Software …* y aparecerá esta ventana emergente:

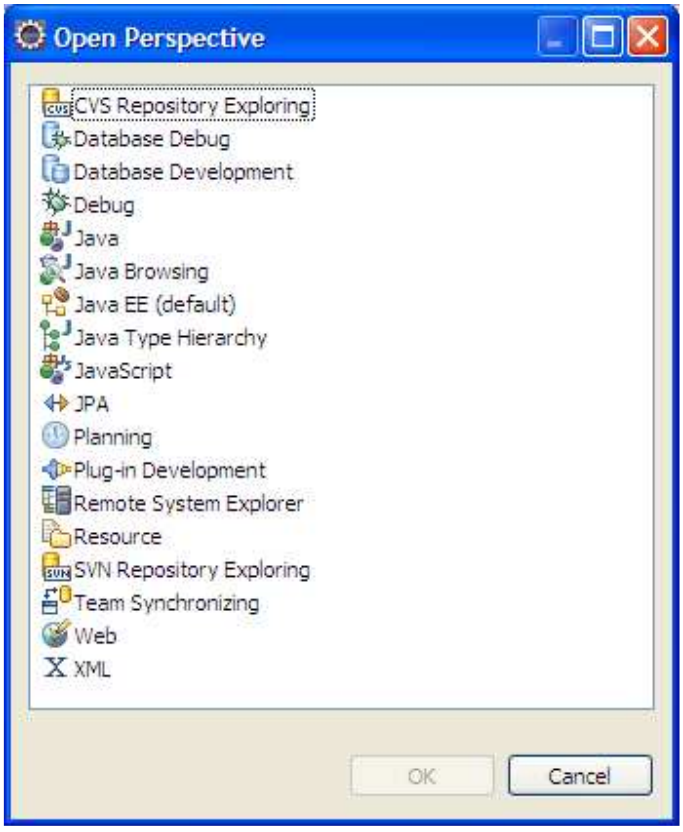


Pinchamos el botón de *Add*, y en la ventana emergente rellenamos los dos campos:



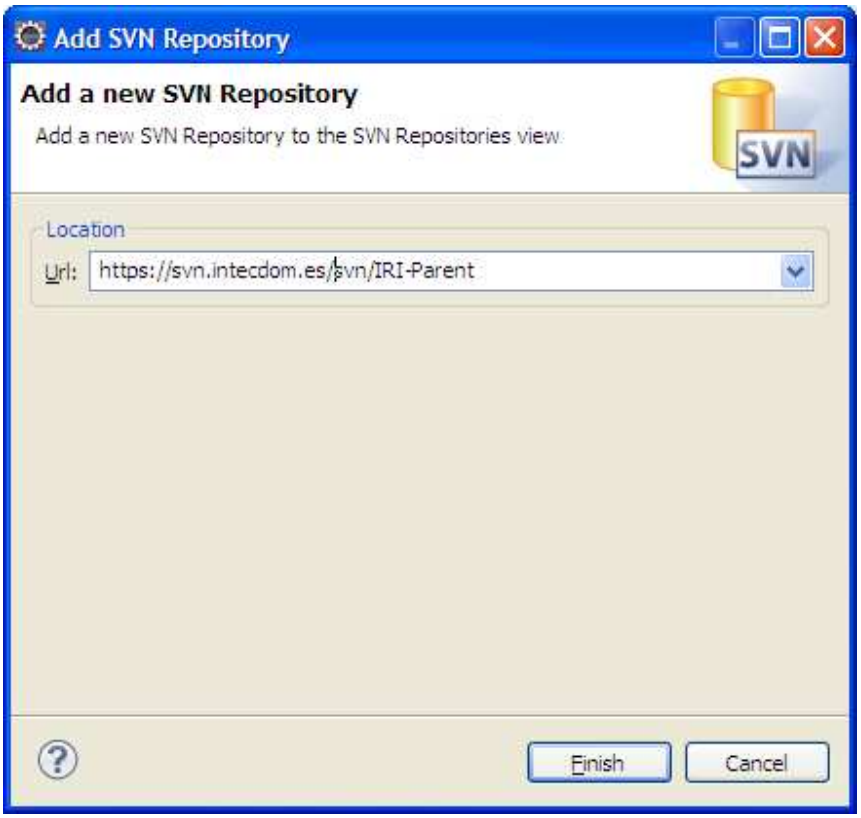
En *Name* ponemos el nombre que prefiramos, en *Location* escribimos la URL correspondiente. Finalmente pulsamos OK y la instalación comenzará.

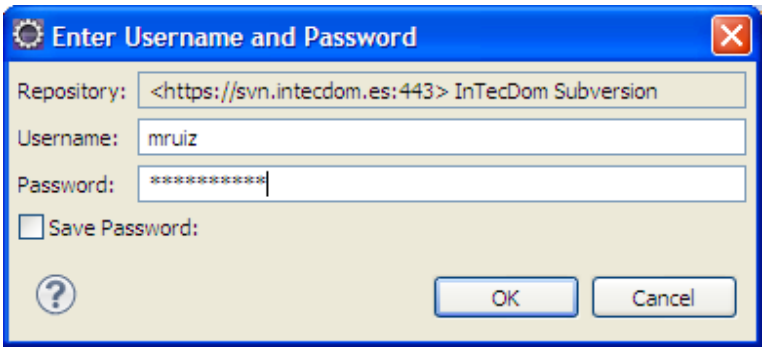
Para descargar el contenido guardado en el repositorio, hay que hacer clic en la opción del Menú *Windows -> Open Perspective -> Other -> SVN Repository Exploring*:



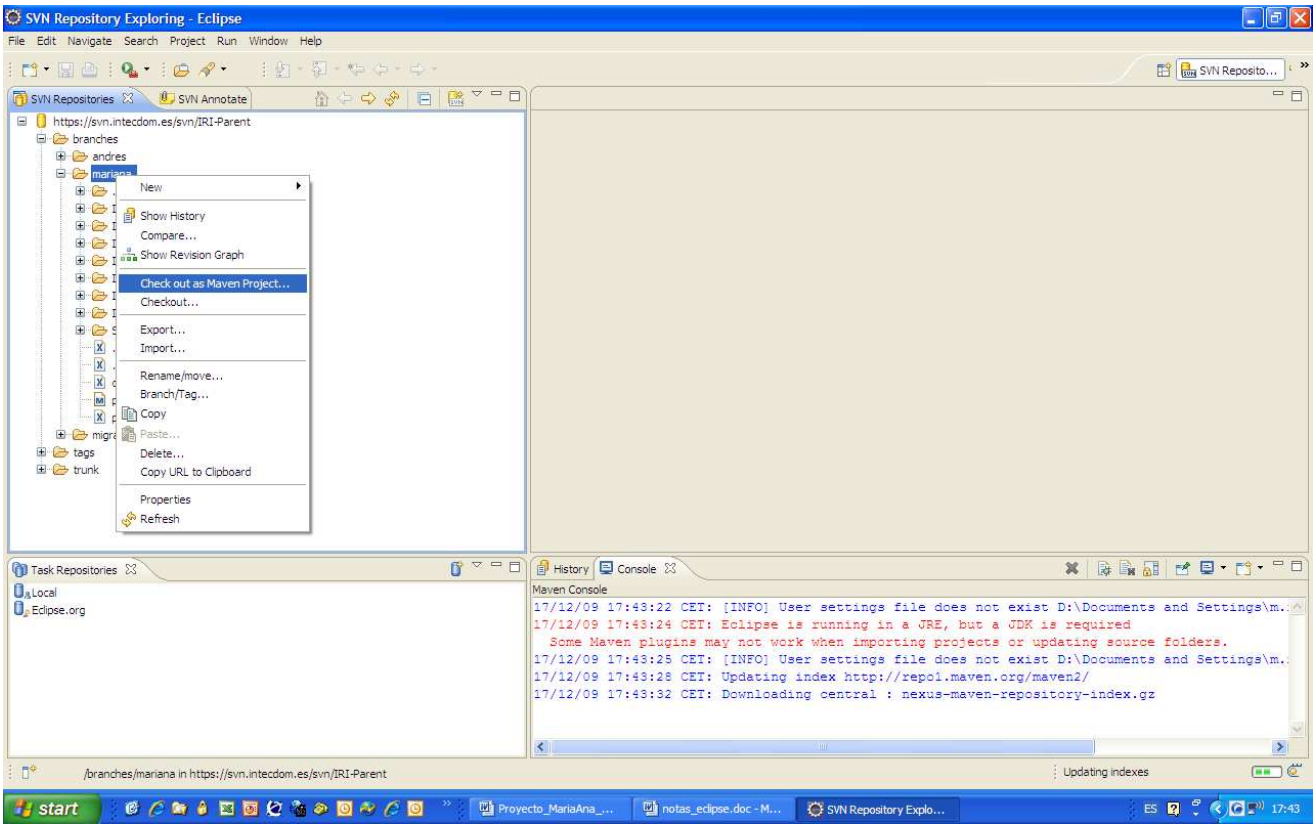
Cuando termine, sobre el espacio en blanco de la pantalla se hace clic en el botón derecho del ratón -> *New -> Repository Location*

Introducimos la URL apropiada, y proporcionamos nuestras credenciales (usuario, password)


Add SVN Repository
Add a new SVN Repository
Add a new SVN Repository to the SVN Repositories view
SVN
Location
Url:
https://svn.intecdom.es/svn/IRI-Parent
Finish
Cancel


Enter Username and Password
Repository:
<https://svn.intecdom.es:443> InTecDom Subversion
Username:
mruiz
Password:
Save Password:
OK
Cancel

Seleccionar la rama correspondiente, hacer clic derecho en el ratón sobre ella, y seleccionar *Check Out as Maven Project* …



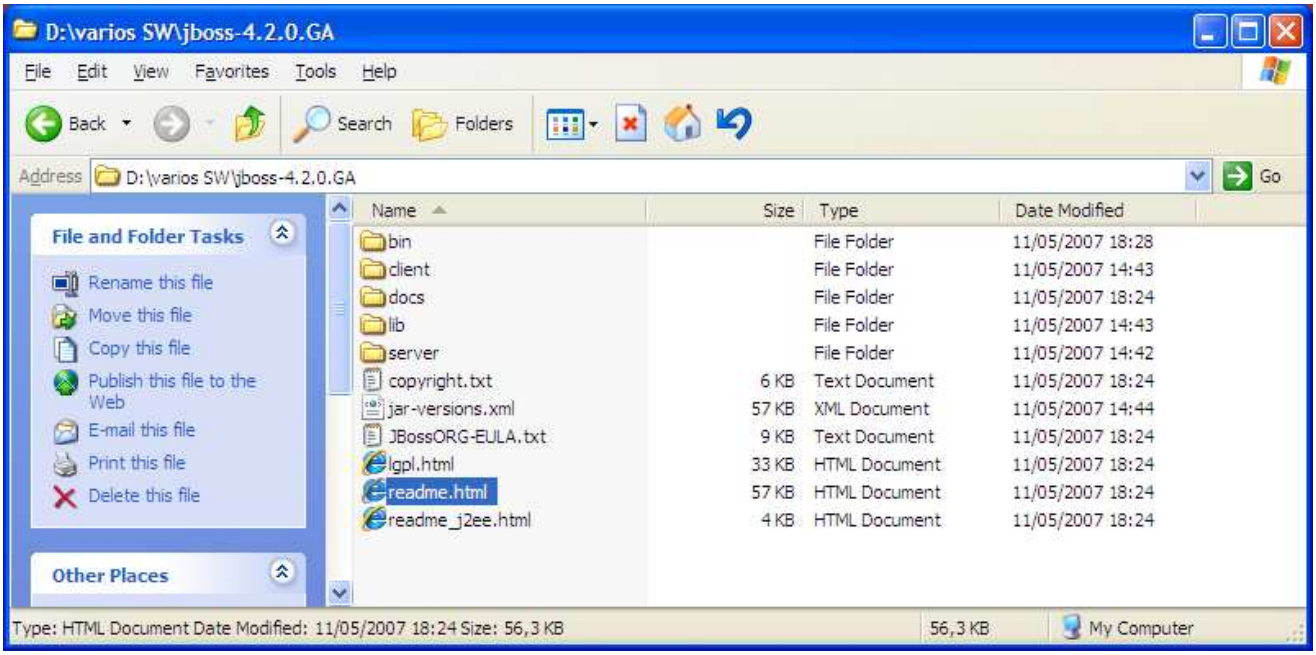
En la ventana emergente que aparecerá -> hacer clic sobre *finish* y se descargará el proyecto desde el repositorio central.

2.2. Descargar e instalar el JBOSS en el PC del desarrollador

Navegamos a la página:
http://www.jboss.org/jbossas/downloads/

Seleccionamos la versión:
4.2.0.GA Stable90 MB 2007-05-11 LGPL 42370

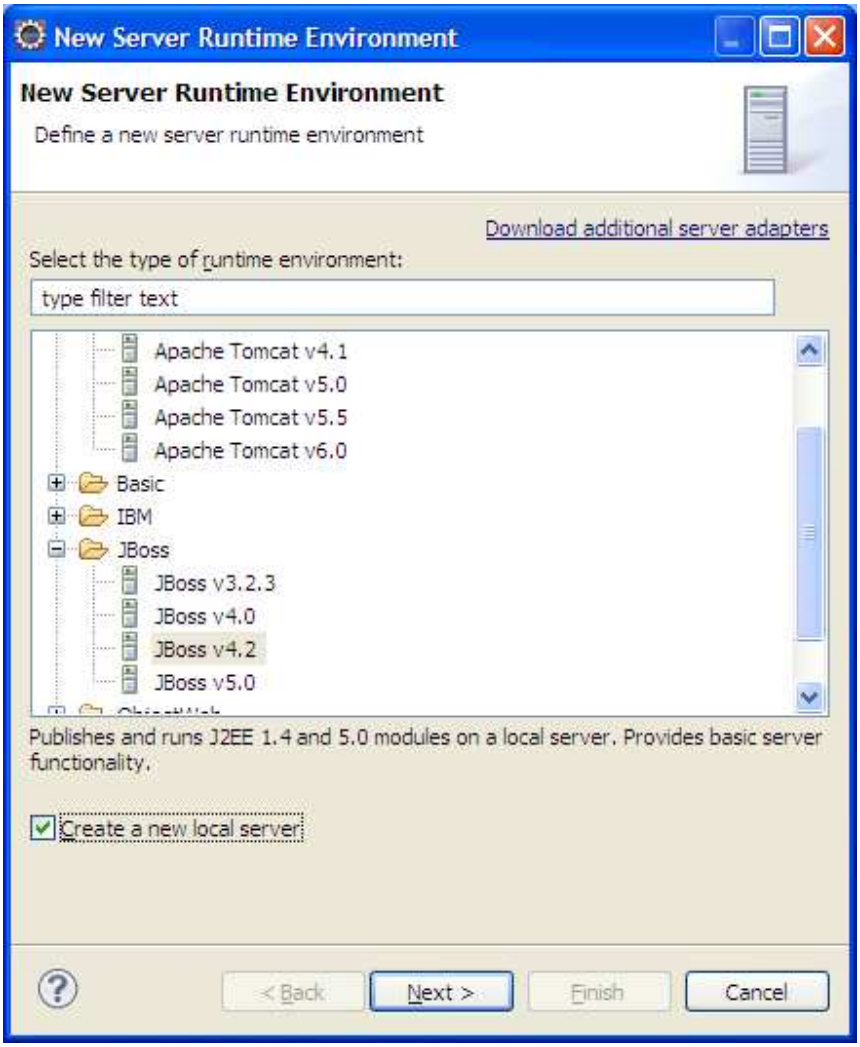
Tras descargar un archivo comprimido ZIP, lo descomprimimos y obtenemos la carpeta correspondiente.



Para añadir la referencia al Jboss en el Eclipse, hacemos clic en la opción de Menú *Windows* > *Prefereces* > *Server* > *Runtime Environmnet* . En la ventana emergente pinchamos *Add*

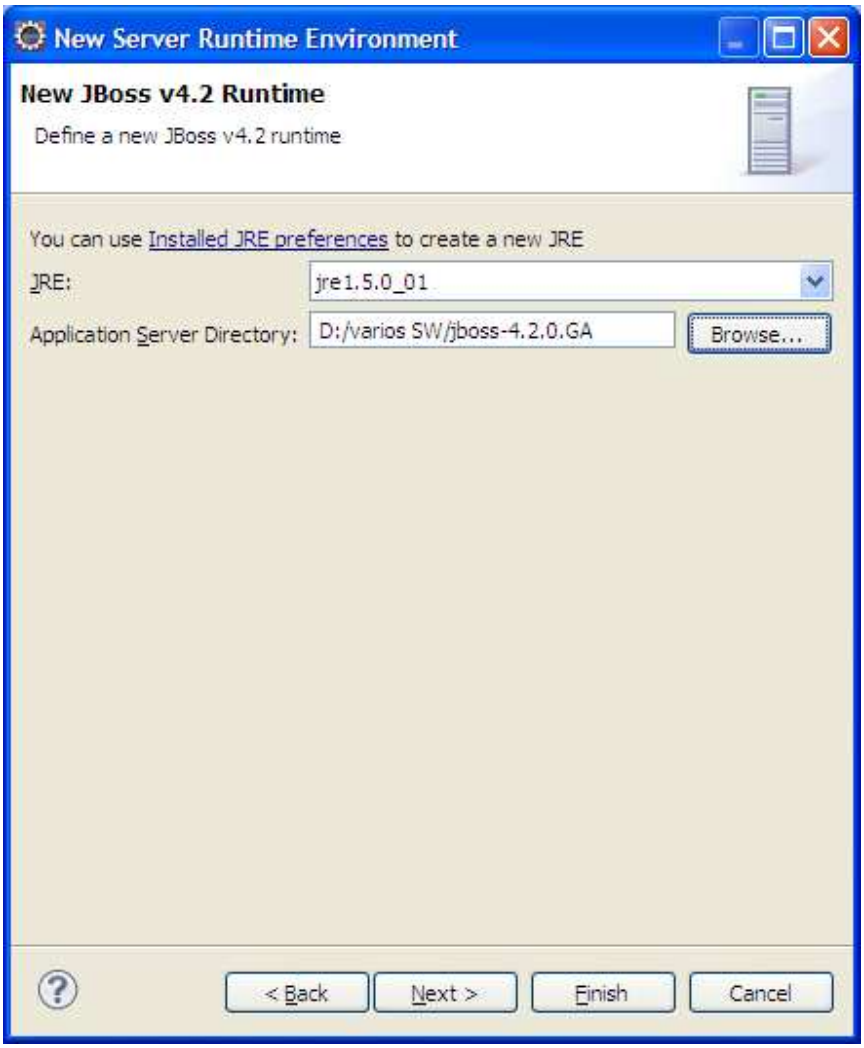
Bajamos por el explorador hasta la entrada del JBoss, lo desplegamos y hacemos clic sobre la opción JBoss v4.2.

En la parte inferior de la ventana, marcamos la opción *Create a new local Server*.

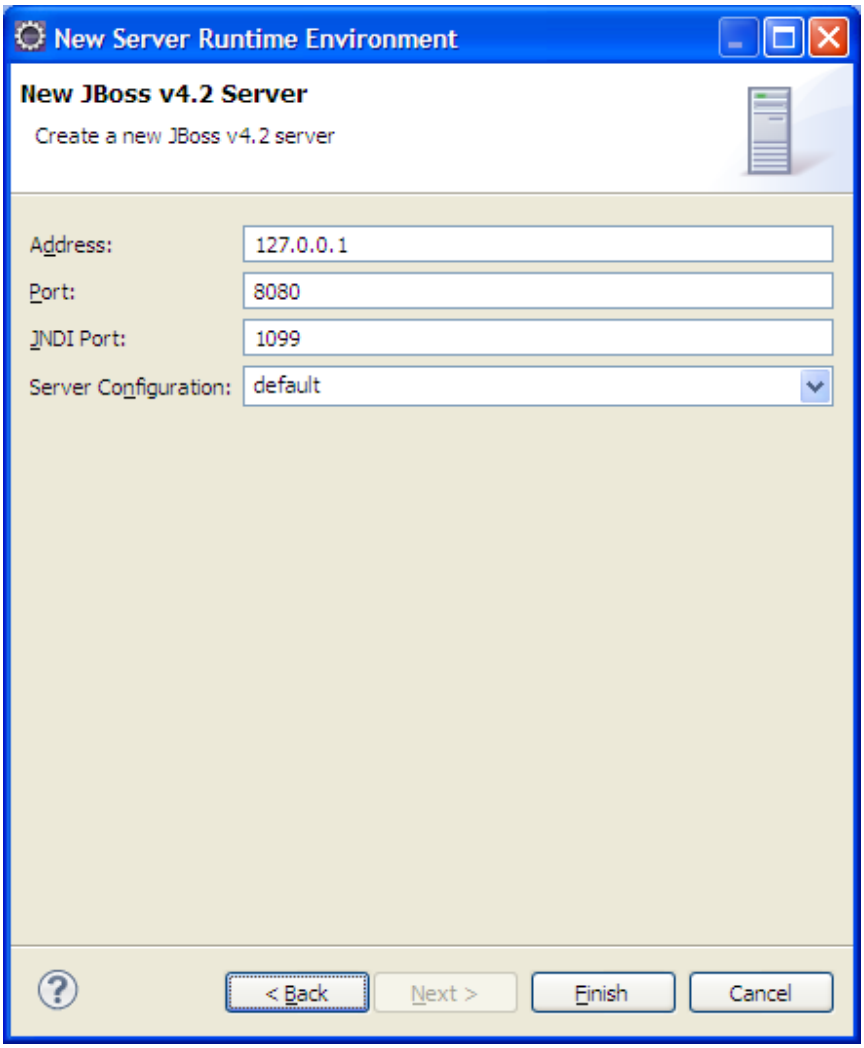


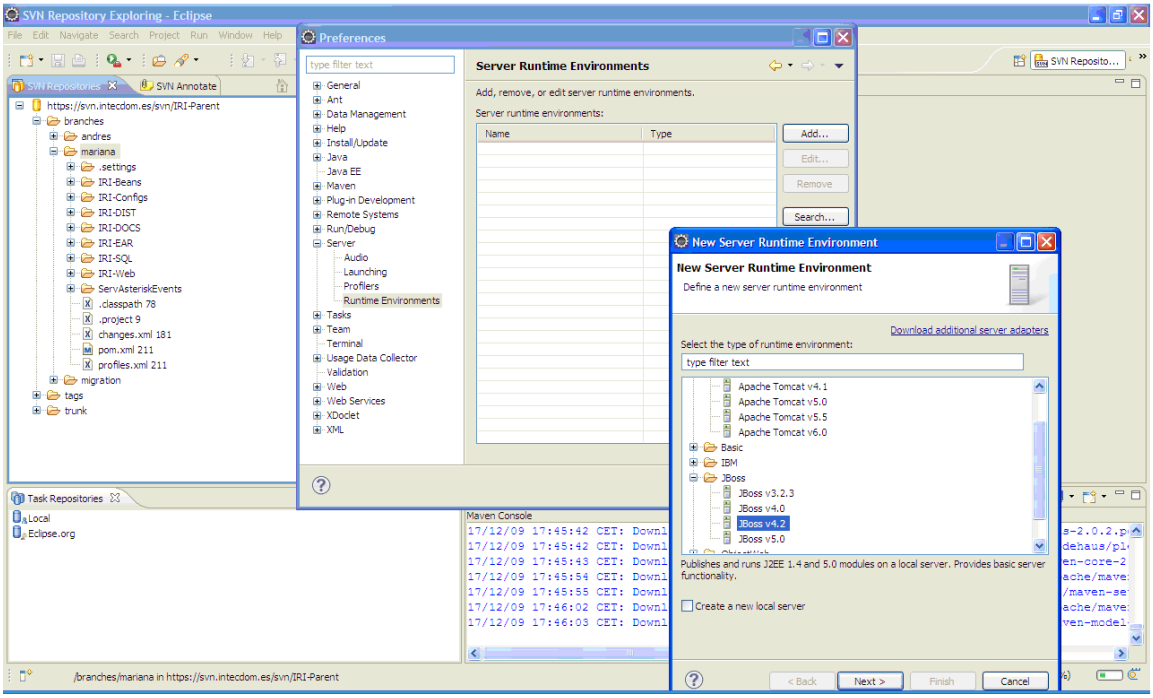
Revisamos los dos campos:

- JRE: se rellena por defecto, comprobamos que es la versión de JRE correcta de nuestro PC
- *Application Server Directory*: hacemos clic sobre el botón *Browse* y seleccionamos el directorio donde hemos creado el directorio del jboss en nuestro PC:



Revisamos los parámetros en la ventana siguiente y pulsamos *Finish*.





Finalmente cerramos la ventana resultante haciendo clic sobre el botón OK.

2.3. Configurar Maven2

Descargamos el fichero de configuración de Maven desde cat /root/.m2/settings.xml, adjunto abajo, y creo el fichero en la carpeta correspondiente de mi PC:

D:\Documents and Settings\m.ruiz.canamero\.m2\settings.xml

```
<settings>

    <profiles>
        <!-- Perfil de conexion al SVN -->
        <profile>
            <id>scmConfig</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
                <scm.user>andres</scm.user>
                <scm.password>asa78045</scm.password>
            </properties>
        </profile>
        <!--Perfil de repositorios locales-->
        <profile>
            <id>Repositorios Locales</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <repositories>
                <repository>
                    <id>itd_releases_repo</id>
                    <url>
                        http://repo.intecdom.es:8080/nexus/content/repositories/releases
                    </url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>false</enabled>
                    </snapshots>
                </repository>
                <repository>
                    <id>itd_snapshots_repo</id>
                    <url>
                        http://repo.intecdom.es:8080/nexus/content/repositories/snapshots
                    </url>
                    <releases>
                        <enabled>false</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
        </profile>
    </profiles>

    <servers>
        <server>
```

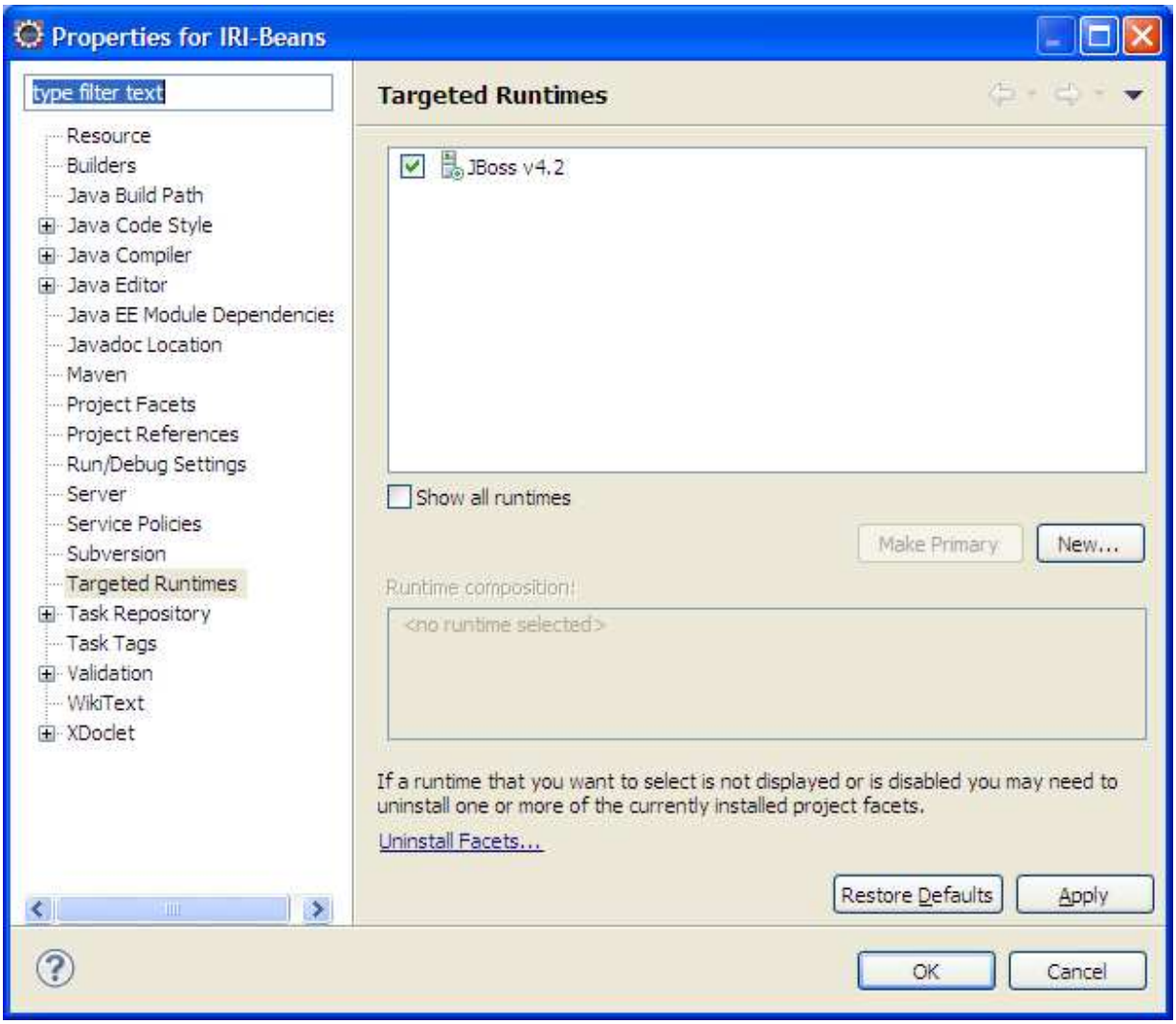
```
            <id>itd_releases_repo</id>
            <username>andres</username>
            <password>a4C$i5A</password>
        </server>
        <server>
            <id>itd_cache_repo</id>
            <username>andres</username>
            <password>a4C$i5A</password>
        </server>
        <server>
            <id>itd_snapshots_repo</id>
            <username>andres</username>
            <password>a4C$i5A</password>
        </server>
        <server>
            <id>website</id>
            <username>maven</username>
            <password>contrasena</password>
            <filePermissions>664</filePermissions>
            <directoryPermissions>775</directoryPermissions>
        </server>
    </servers>

    <!-- Activamos nuestro Nexus como repositorio remoto por defecto-->
    <mirrors>
        <mirror>
            <id>itd_cache_repo</id>
            <url>http://repo.intecdom.es:8080/nexus/content/groups/public
            </url>
            <mirrorOf>*</mirrorOf>
            <name>Repositorio cache ITD.</name>
        </mirror>
    </mirrors>
</settings>
```

Descargamos las dependencias de Maven.

Asociamos todas las carpetas al jboss:

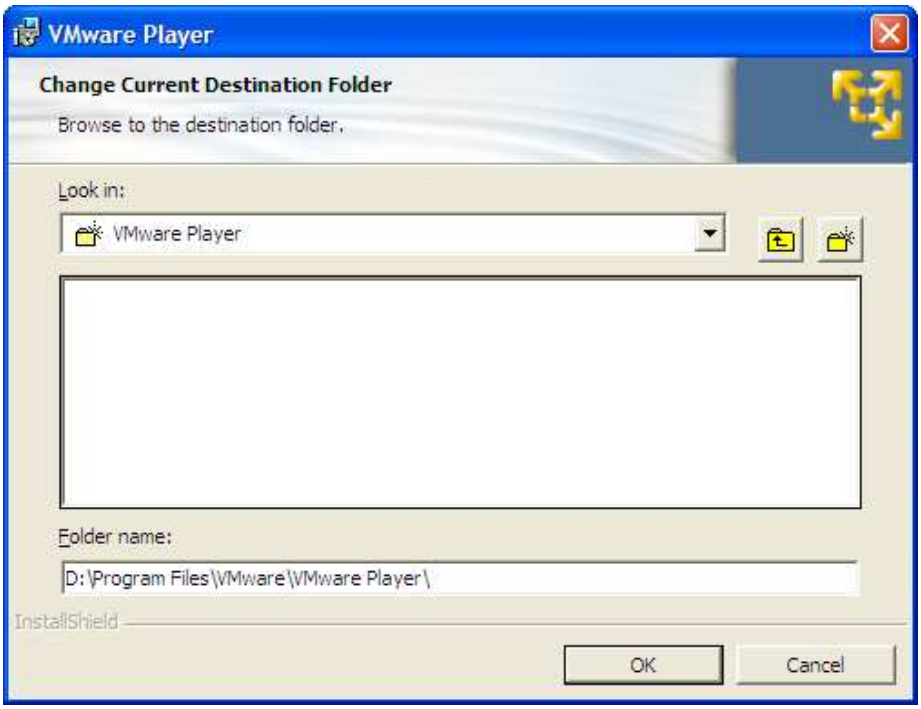
Desde el explorador, selecciono la primera carpeta IRI-Beans, clic derecho > *Properties* > *Targeted Runtimes* y selecciono la opción de JBoss:

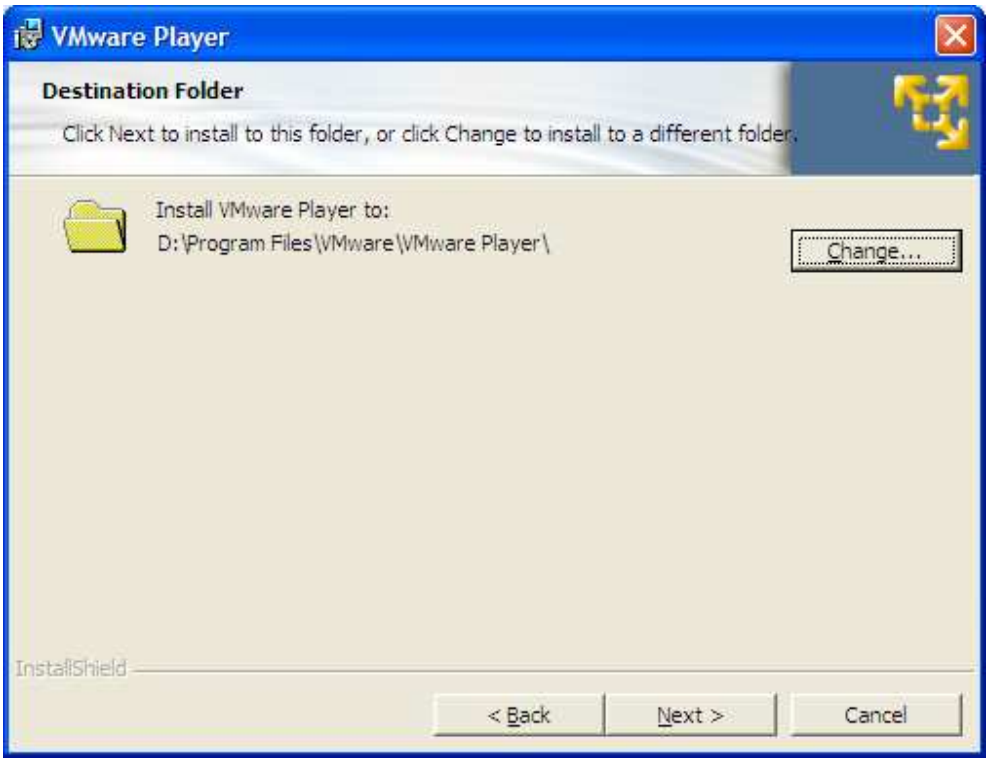


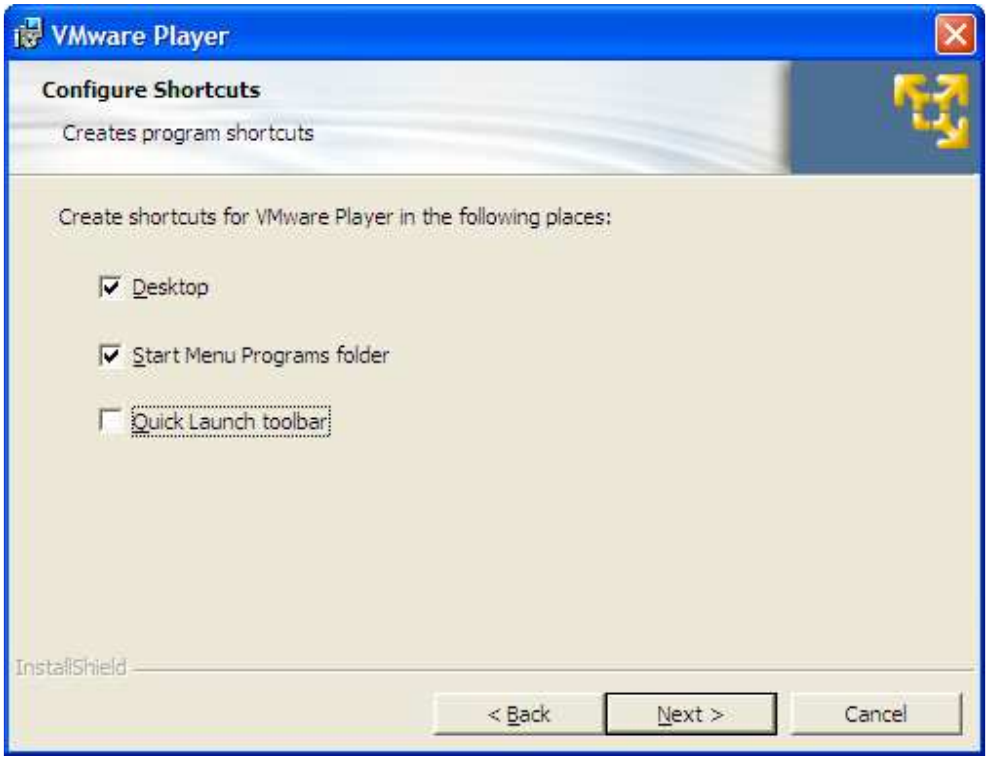
Repito lo mismo para todas las demás carpetas.

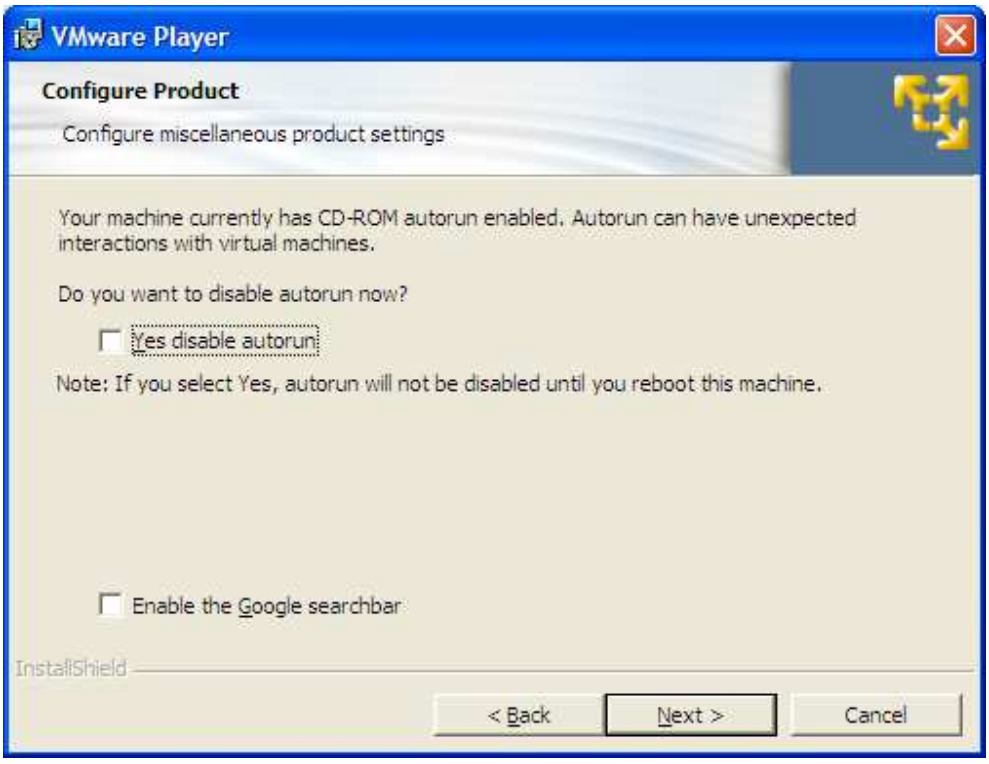
3. Manual de instalación de VMware Player, Debian y OpenLDAP

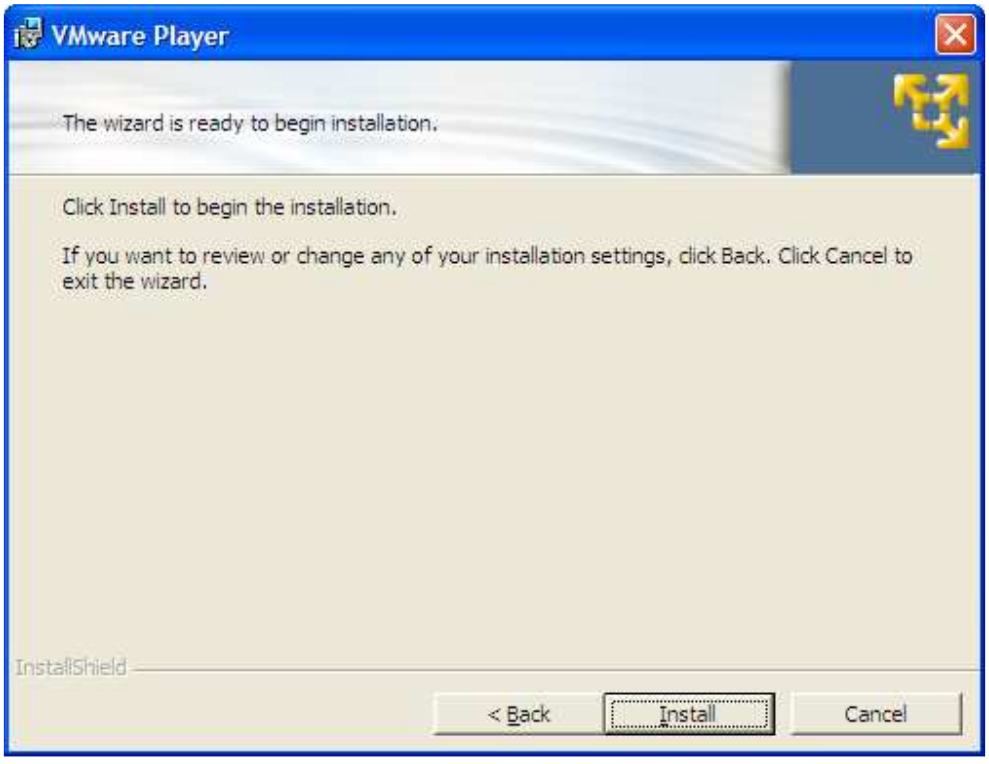
Instalación sobre Windows en el PC del desarrollador para hacer las pruebas iniciales.

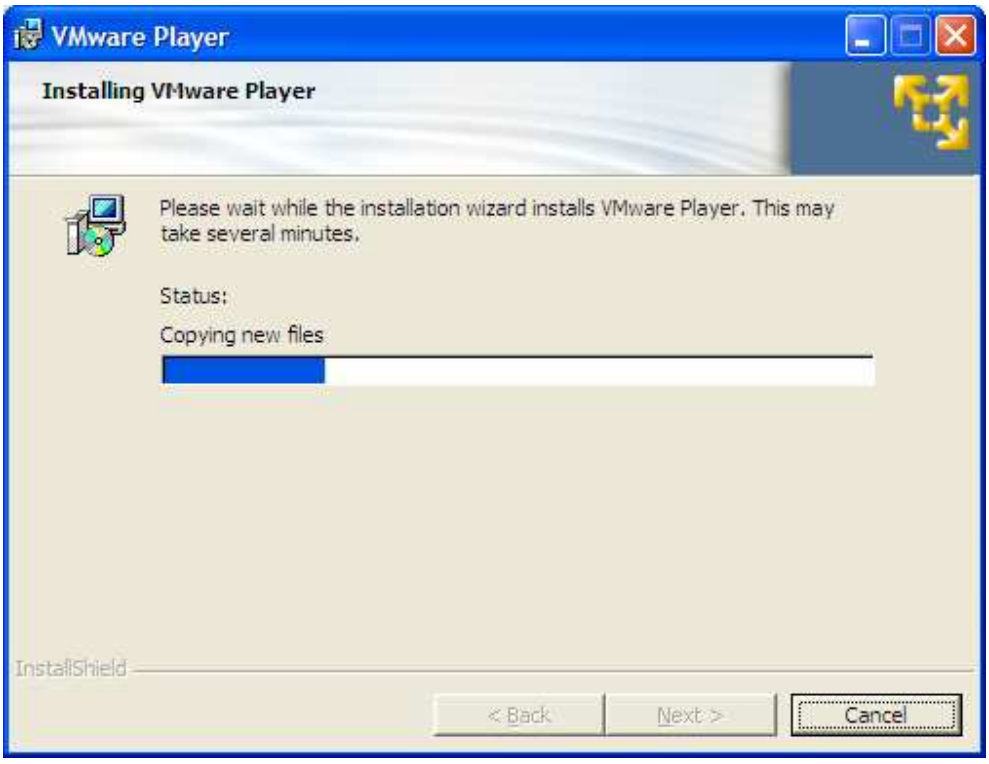


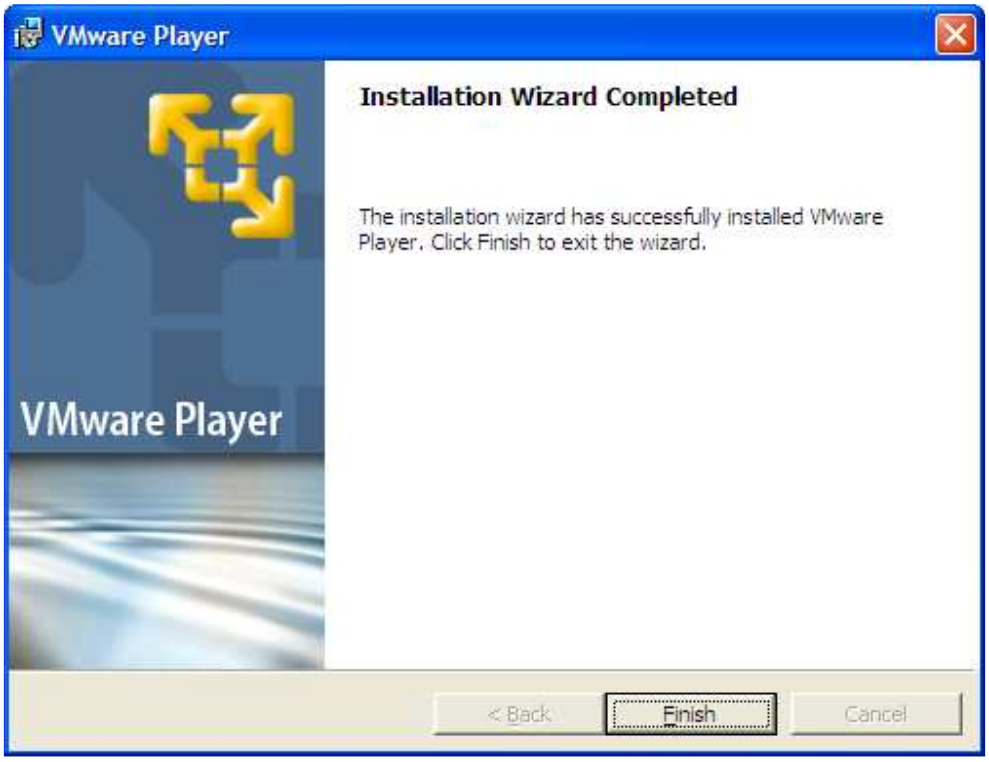

VMware Player
Destination Folder
Click Next to install to this folder, or click Change to install to a different folder.
Install VMware Player to:
D:\Program Files\VMware\VMware Player\
Change...
InstallShield
< Back
Next >
Cancel

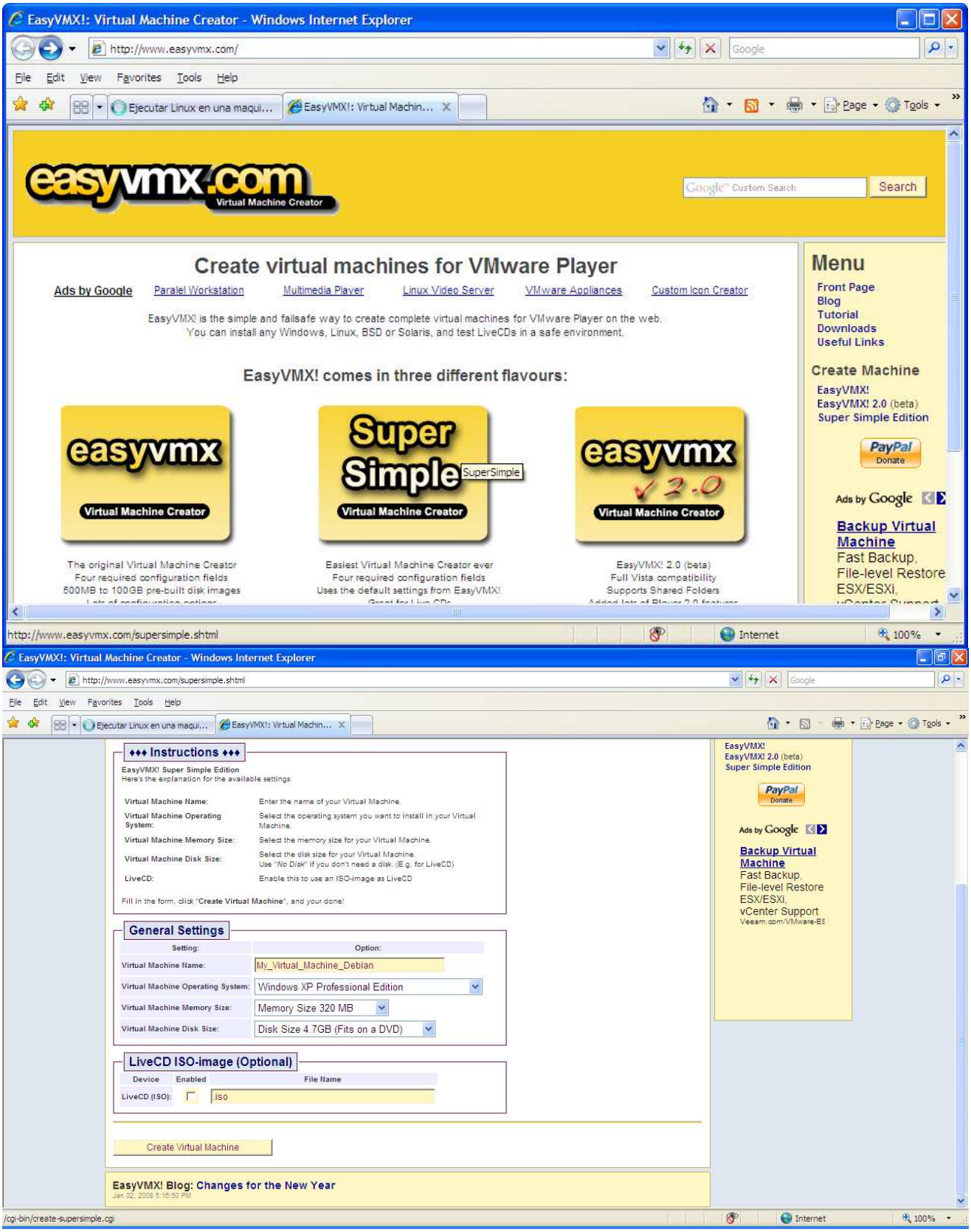

VMware Player
Configure Shortcuts
Creates program shortcuts
Create shortcuts for VMware Player in the following places:
Desktop
Start Menu Programs folder
Quick Launch toolbar
InstallShield
< Back
Next >
Cancel

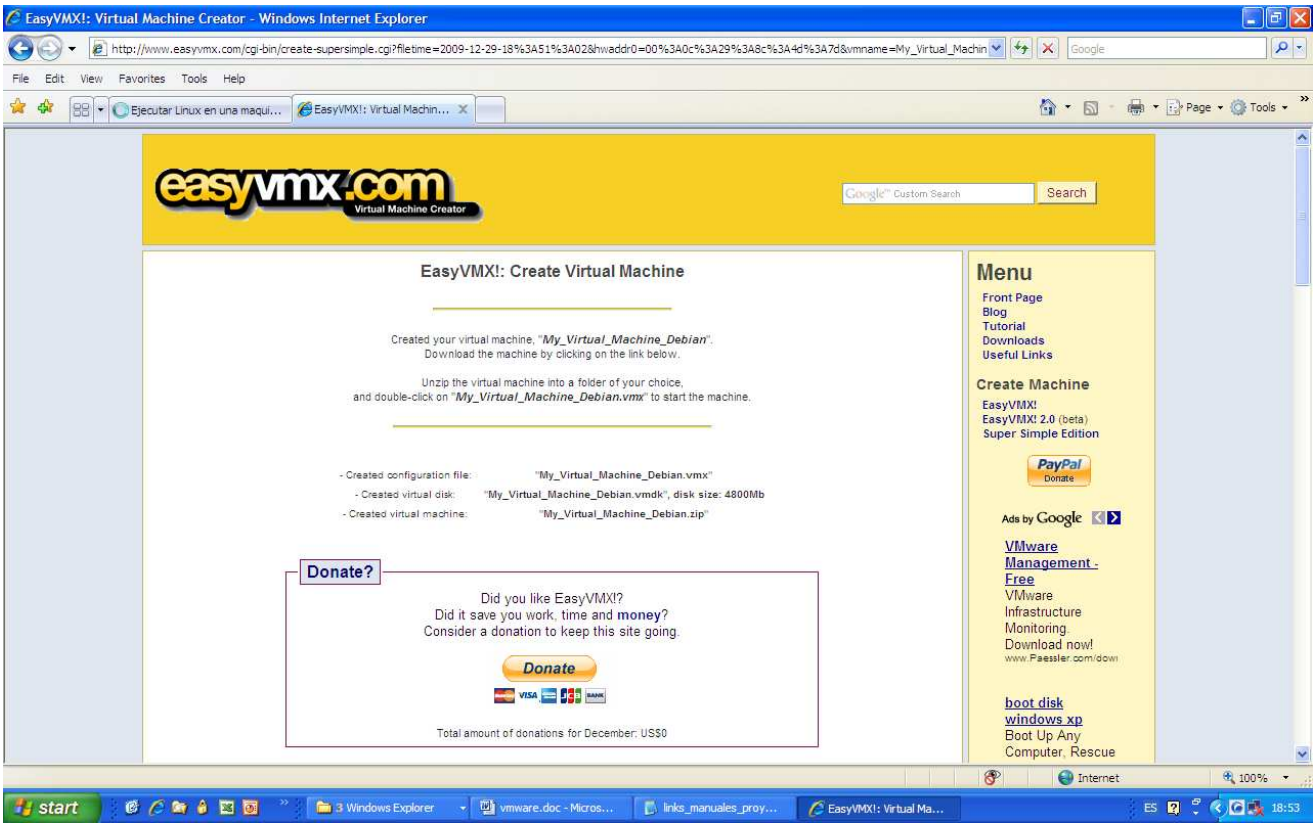

VMware Player
Configure Product
Configure miscellaneous product settings
Your machine currently has CD-ROM autorun enabled. Autorun can have unexpected interactions with virtual machines.
Do you want to disable autorun now?
Yes disable autorun
Note: If you select Yes, autorun will not be disabled until you reboot this machine.
Enable the Google searchbar
InstallShield
< Back
Next >
Cancel


VMware Player
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield
< Back
Install
Cancel

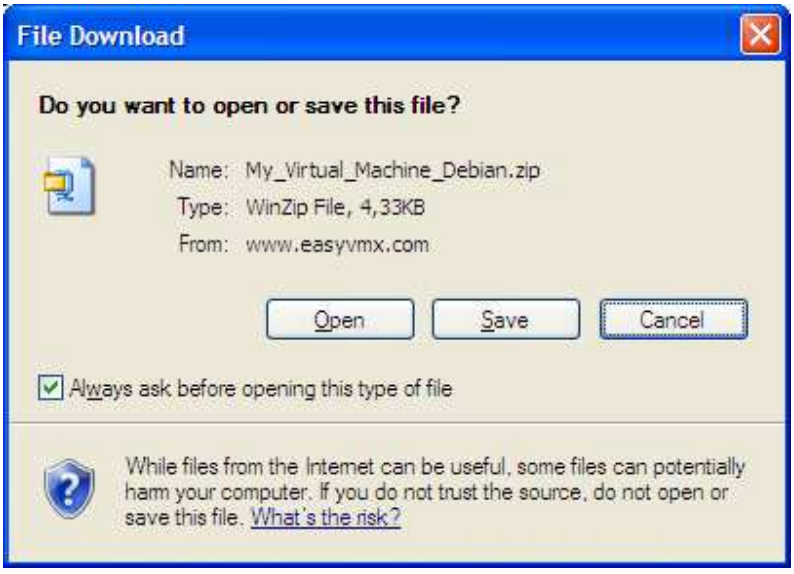

VMware Player
Installing VMware Player
Please wait while the installation wizard installs VMware Player. This may take several minutes.
Status:
Copying new files
InstallShield
< Back
Next >
Cancel


VMware Player
VMware Player
Installation Wizard Completed
The installation wizard has successfully installed VMware Player. Click Finish to exit the wizard.
< Back
Finish
Cancel

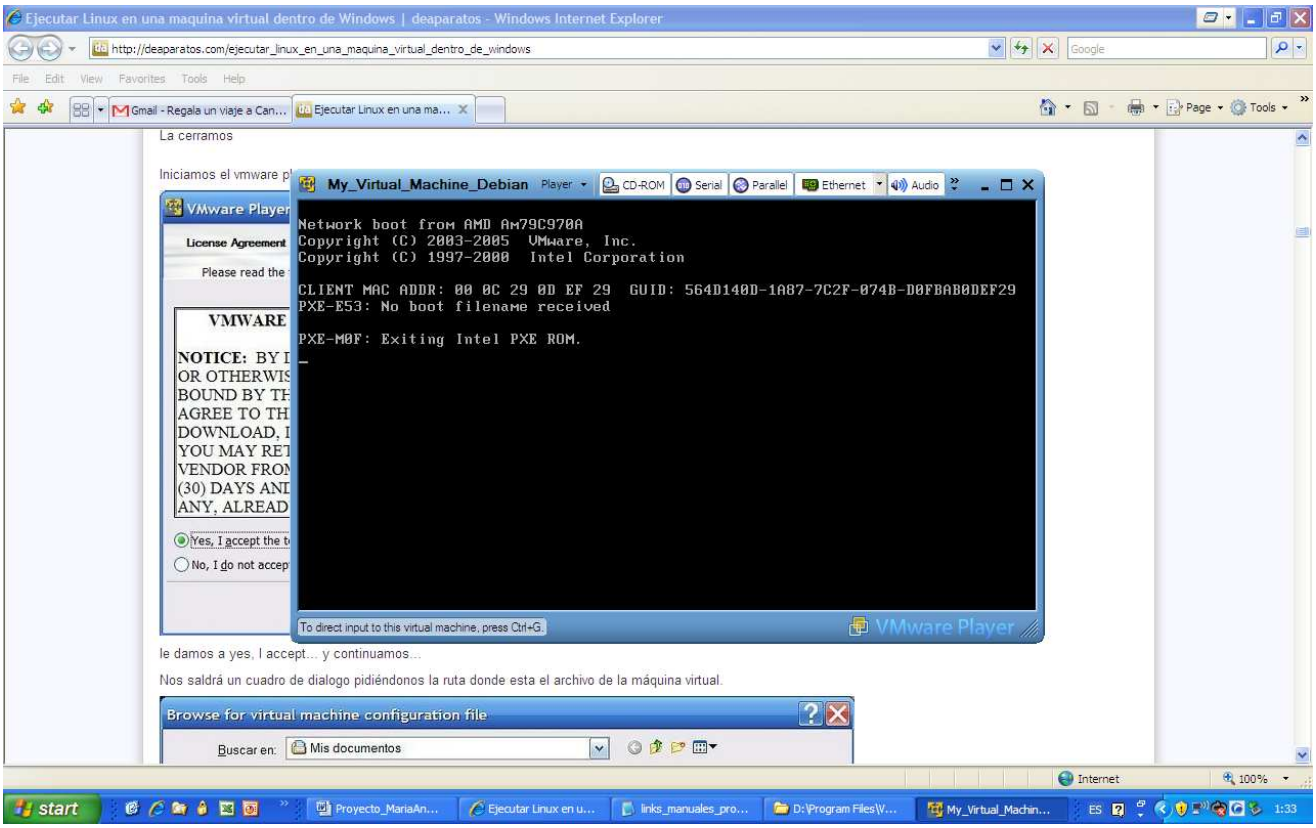

EasyVMX!: Virtual Machine Creator - Windows Internet Explorer
http://www.easyvmx.com/
File Edit View Favorites Tools Help
Ejecutar Linux en una maqui...
EasyVMX!: Virtual Machin...
Page Tools
easyvmx.com
Virtual Machine Creator
Google Custom Search
Search
Create virtual machines for VMware Player
Ads by Google
Parallel Workstation
Multimedia Player
Linux Video Server
VMware Appliances
Custom Icon Creator
EasyVMX! is the simple and failsafe way to create complete virtual machines for VMware Player on the web.
You can install any Windows, Linux, BSD or Solaris, and test LiveCDs in a safe environment.
EasyVMX! comes in three different flavours:
easyvmx
Virtual Machine Creator
Super Simple
SuperSimple
Virtual Machine Creator
easyvmx
v2.0
Virtual Machine Creator
The original Virtual Machine Creator
Four required configuration fields
500MB to 100GB pre-built disk images
Easiest Virtual Machine Creator ever
Four required configuration fields
Uses the default settings from EasyVMX!
EasyVMX! 2.0 (beta)
Full Vista compatibility
Supports Shared Folders
Menu
Front Page
Blog
Tutorial
Downloads
Useful Links
Create Machine
EasyVMX!
EasyVMX! 2.0 (beta)
Super Simple Edition
PayPal Donate
Ads by Google
Backup Virtual Machine
Fast Backup,
File-level Restore
ESX/ESXi,
http://www.easyvmx.com/supersimple.shtml
Internet
100%
EasyVMX!: Virtual Machine Creator - Windows Internet Explorer
http://www.easyvmx.com/supersimple.shtml
♦♦♦ Instructions ♦♦♦
EasyVMX! Super Simple Edition
Here's the explanation for the available settings:
Virtual Machine Name: Enter the name of your Virtual Machine.
Virtual Machine Operating System: Select the operating system you want to install in your Virtual Machine.
Virtual Machine Memory Size: Select the memory size for your Virtual Machine.
Virtual Machine Disk Size: Select the disk size for your Virtual Machine. Use "No Disk" if you don't need a disk. (E.g. for LiveCD)
LiveCD: Enable this to use an ISO-image as LiveCD
Fill in the form, click "Create Virtual Machine", and your done!
General Settings
Setting: Option:
Virtual Machine Name: My_Virtual_Machine_Debian
Virtual Machine Operating System: Windows XP Professional Edition
Virtual Machine Memory Size: Memory Size 320 MB
Virtual Machine Disk Size: Disk Size 4.7GB (Fits on a DVD)
LiveCD ISO-image (Optional)
Device Enabled File Name
LiveCD (ISO): .iso
Create Virtual Machine
EasyVMX! Blog: Changes for the New Year
Jan 02, 2008 5:16:50 PM
/cgi-bin/create-supersimple.cgi
Internet
100%

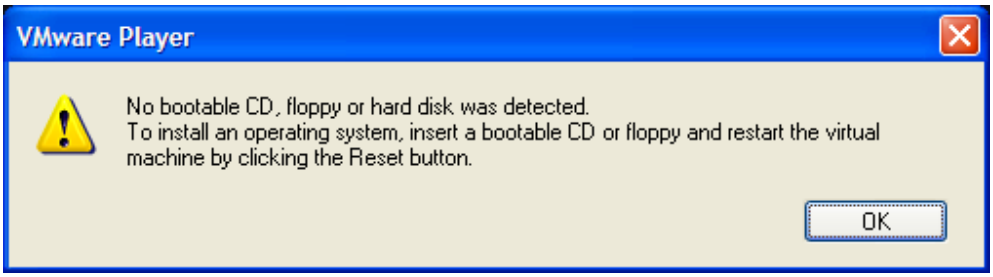


En ella le daremos en Download your virtual machine a Ubuntu.zip y nos bajaremos la maquina virtual que acabamos de crear:

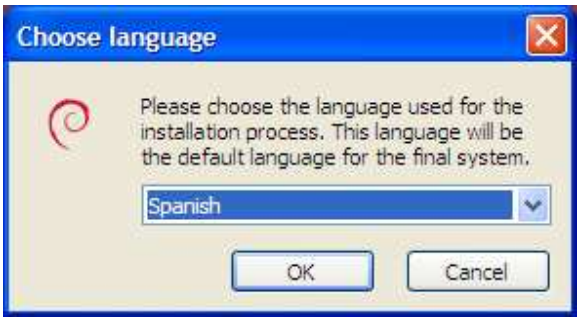


vamos a la carpeta donde hemos bajado el software de vmware, descomprimimos el ZIP y buscamos el fichero .vmx y hacemos clic sobre él:

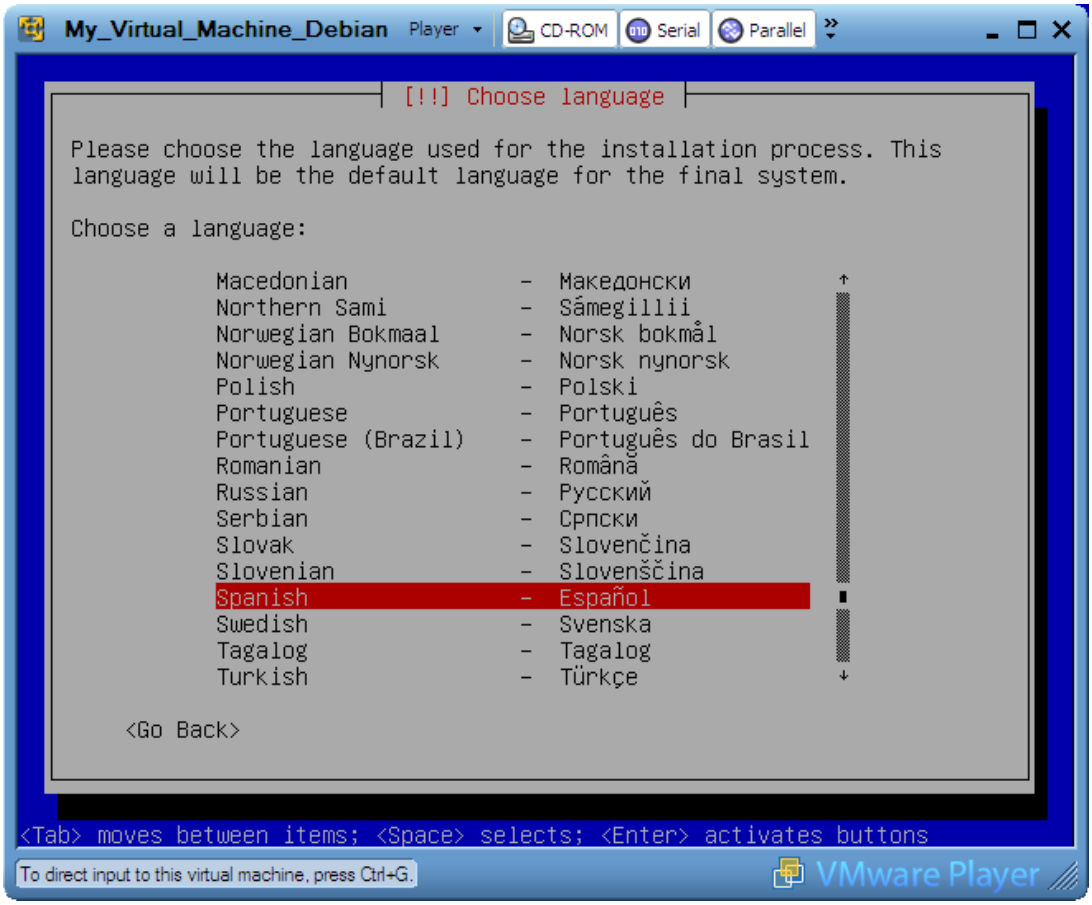


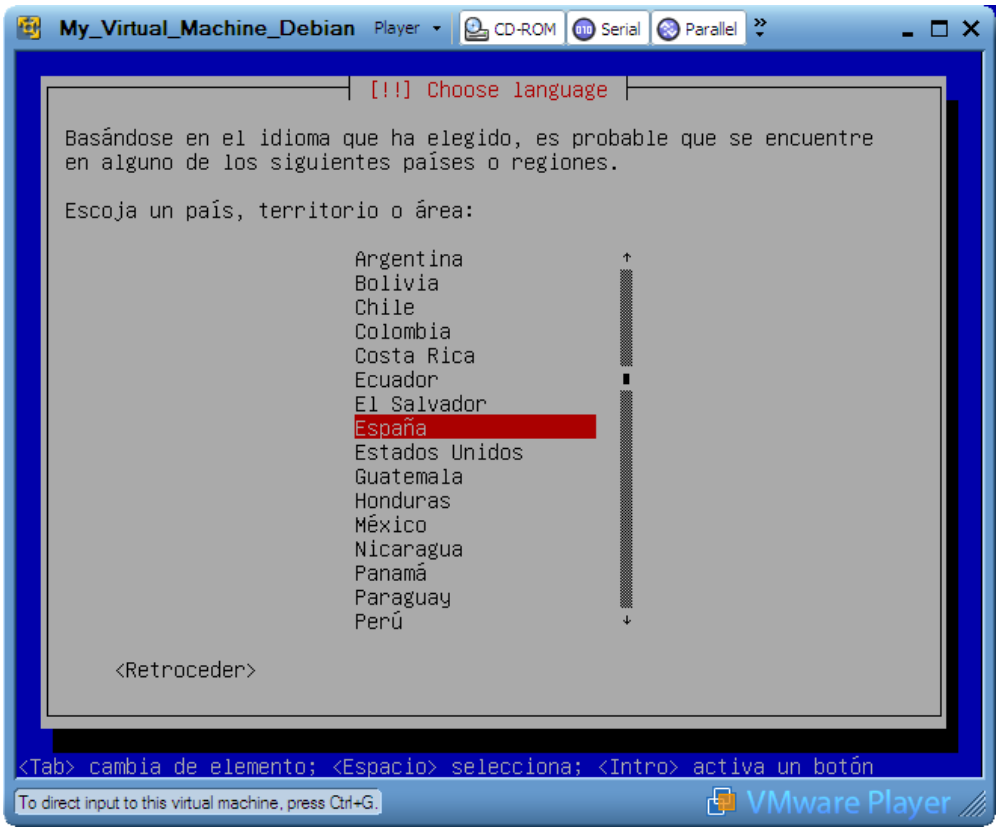


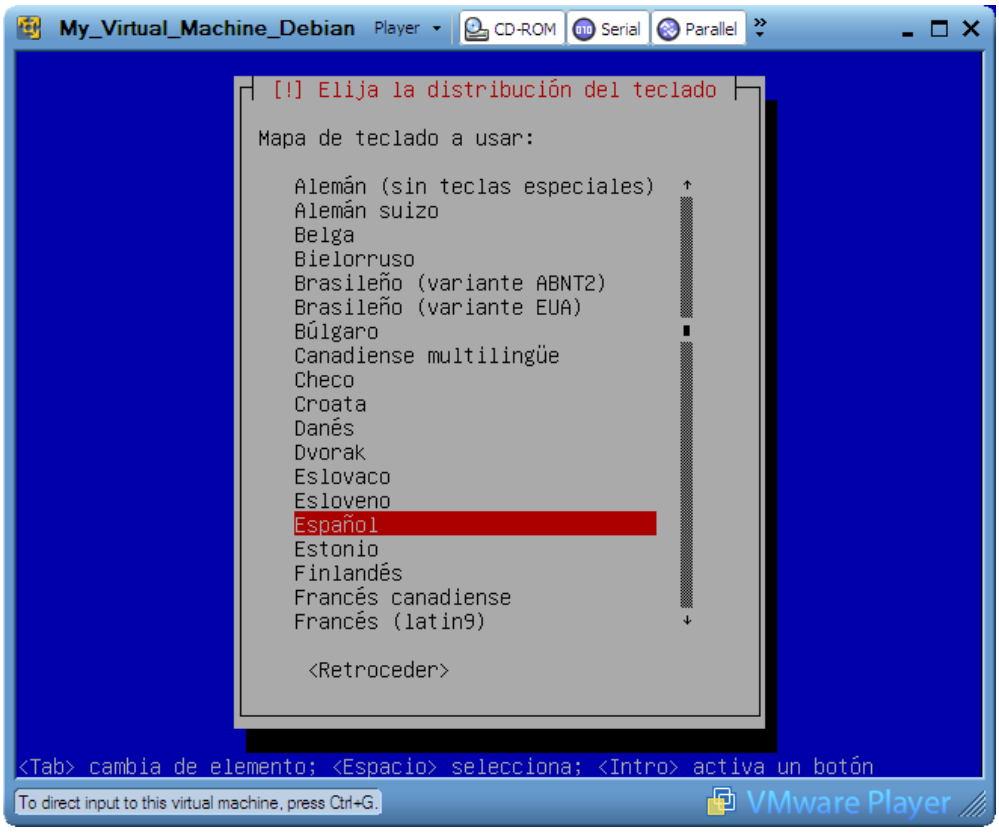
Introducir CD de instalación de Debian

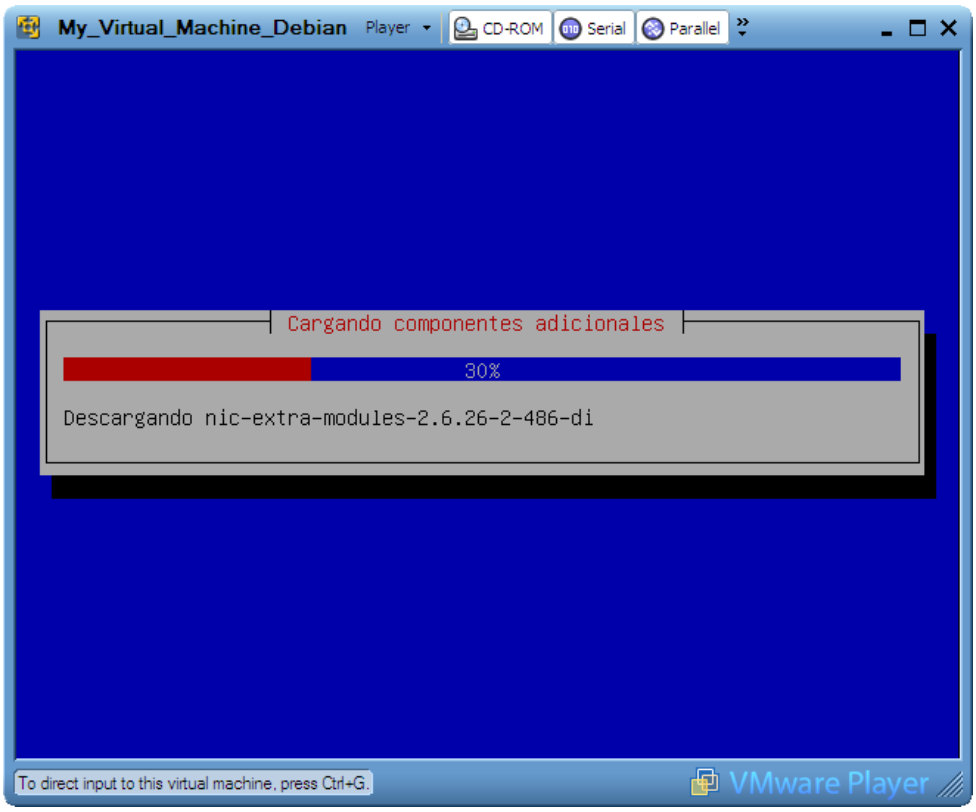


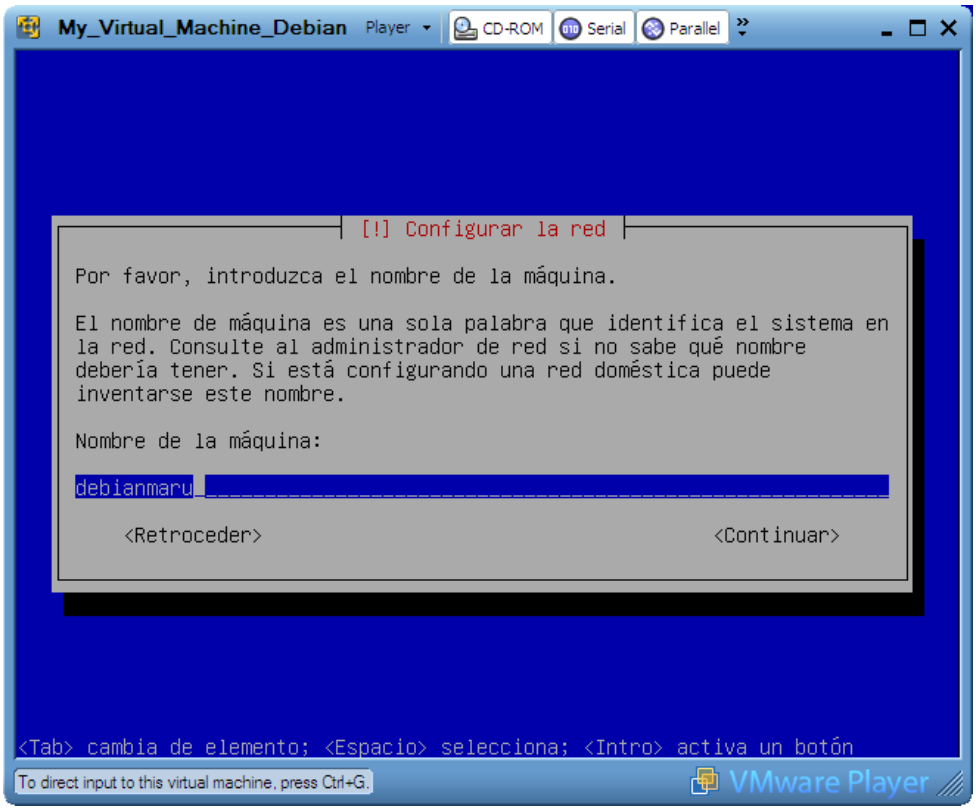

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Installer boot menu
Install
Graphical install
Advanced options
Help
debian
GNU/Linux
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Choose language
Please choose the language used for the installation process. This language will be the default language for the final system.
Choose a language:
Macedonian - Македонски
Northern Sami - Sámegillii
Norwegian Bokmaal - Norsk bokmål
Norwegian Nynorsk - Norsk nynorsk
Polish - Polski
Portuguese - Português
Portuguese (Brazil) - Português do Brasil
Romanian - Română
Russian - Русский
Serbian - Српски
Slovak - Slovenčina
Slovenian - Slovenščina
Spanish - Español
Swedish - Svenska
Tagalog - Tagalog
Turkish - Türkçe
<Go Back>
<Tab> moves between items; <Space> selects; <Enter> activates buttons
To direct input to this virtual machine, press Ctrl+G.
VMware Player

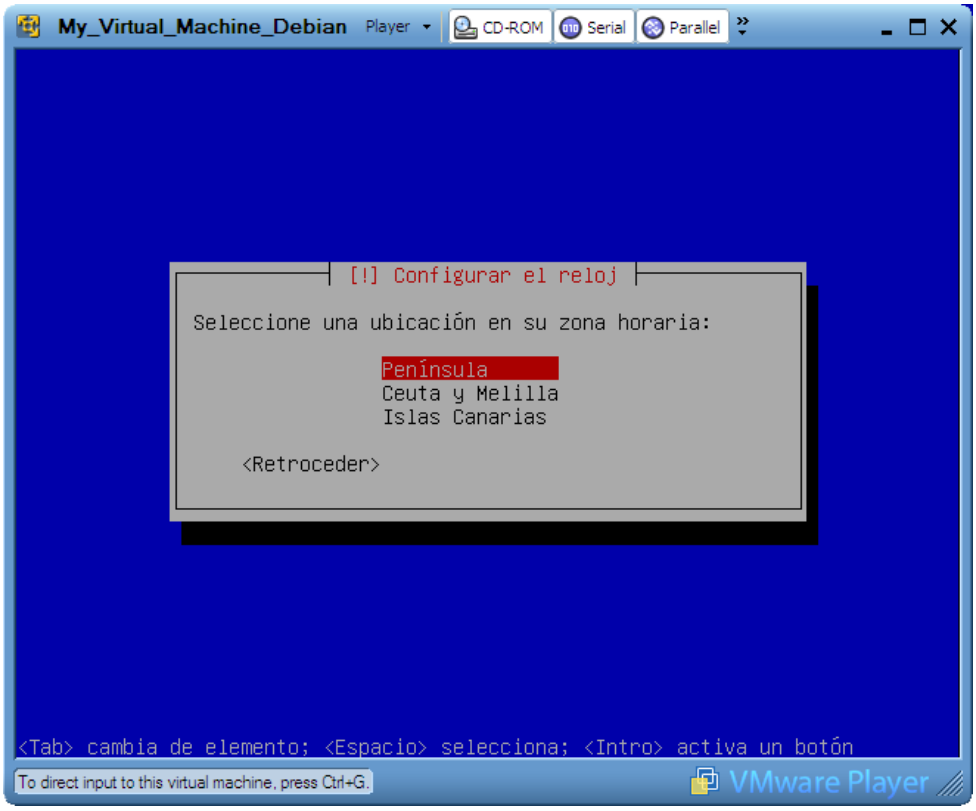

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Choose language
Basándose en el idioma que ha elegido, es probable que se encuentre en alguno de los siguientes países o regiones.
Escoja un país, territorio o área:
Argentina
Bolivia
Chile
Colombia
Costa Rica
Ecuador
El Salvador
España
Estados Unidos
Guatemala
Honduras
México
Nicaragua
Panamá
Paraguay
Perú
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

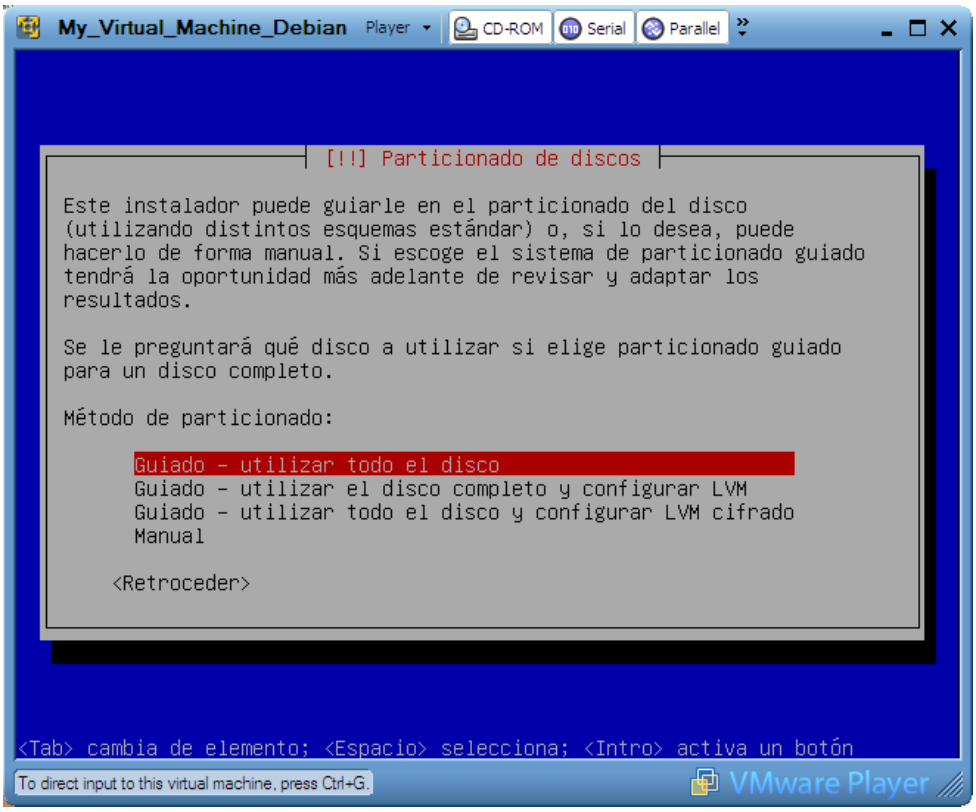

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Elija la distribución del teclado
Mapa de teclado a usar:
Alemán (sin teclas especiales)
Alemán suizo
Belga
Bielorruso
Brasileño (variante ABNT2)
Brasileño (variante EUA)
Búlgaro
Canadiense multilingüe
Checo
Croata
Danés
Dvorak
Eslovaco
Esloveno
Español
Estonio
Finlandés
Francés canadiense
Francés (latin9)
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

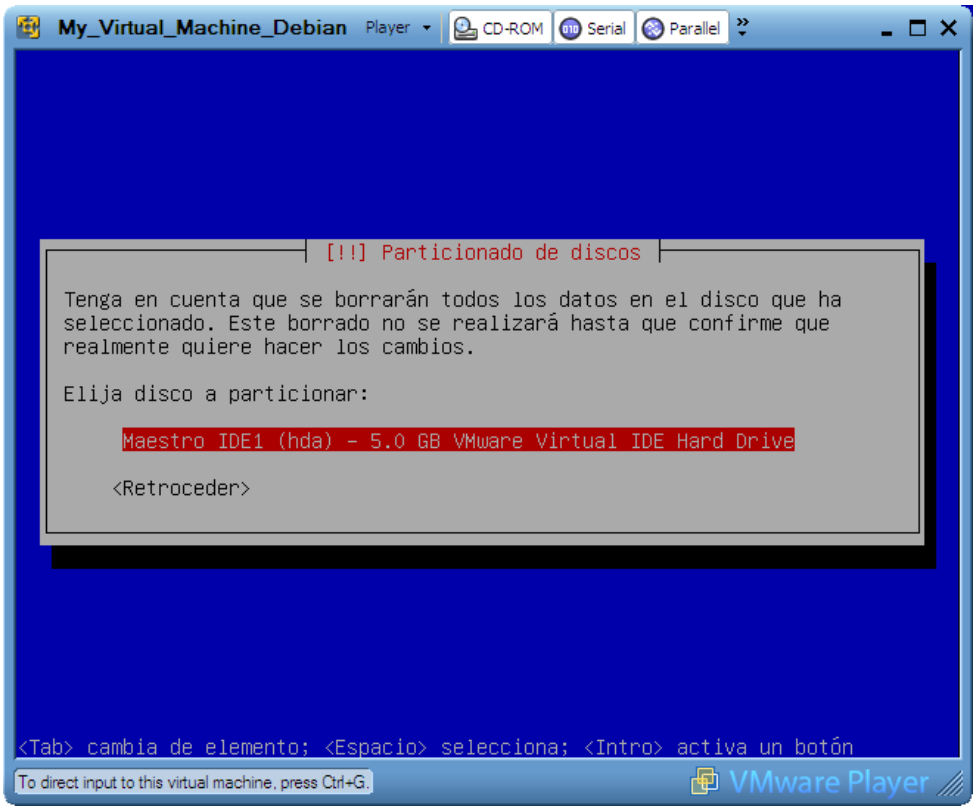

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Cargando componentes adicionales
30%
Descargando nic-extra-modules-2.6.26-2-486-di
To direct input to this virtual machine, press Ctrl+G.
VMware Player

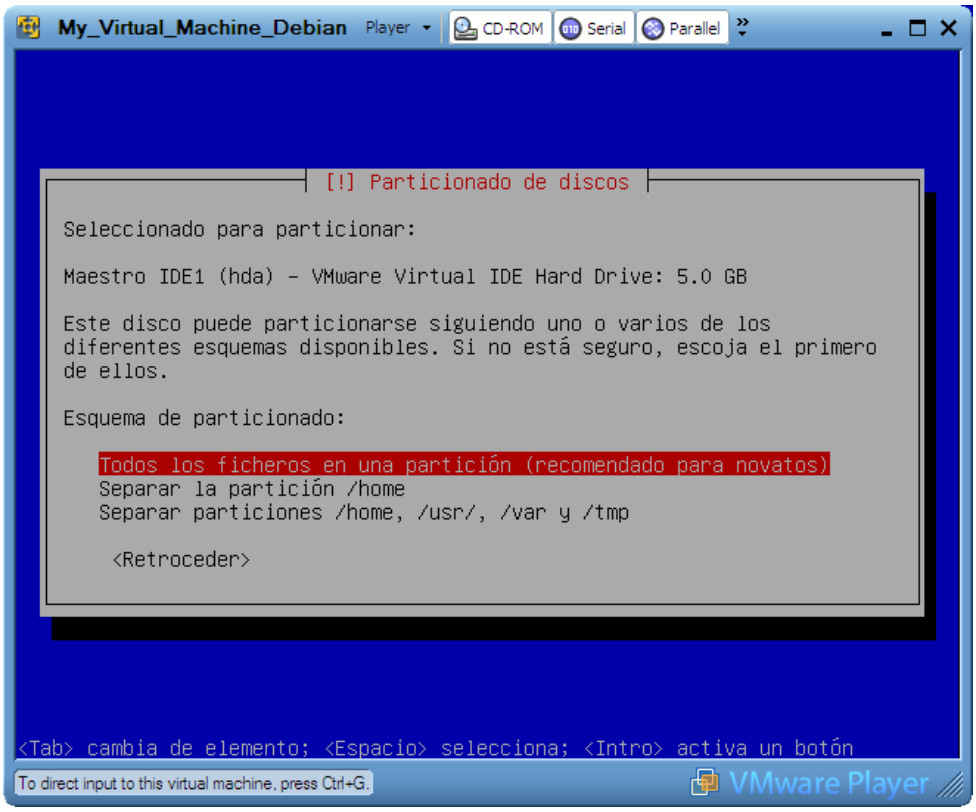

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar la red
Por favor, introduzca el nombre de la máquina.
El nombre de máquina es una sola palabra que identifica el sistema en la red. Consulte al administrador de red si no sabe qué nombre debería tener. Si está configurando una red doméstica puede inventarse este nombre.
Nombre de la máquina:
debianmaru
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

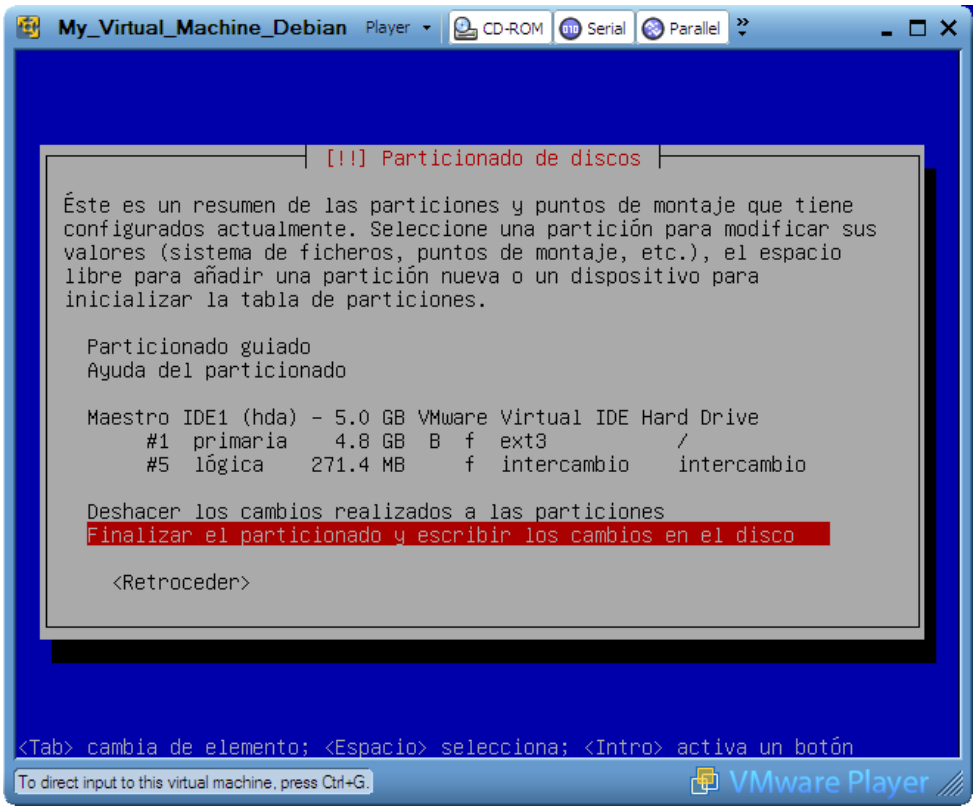

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar la red
El nombre de dominio es la parte de su dirección de Internet a la derecha del nombre de sistema. Habitualmente es algo que termina por .com, .net, .edu, o .org. Puede inventárselo si está instalando una red doméstica, pero asegúrese de utilizar el mismo nombre de dominio en todos sus ordenadores.
Nombre de dominio:
intecdom.es
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar el reloj
Seleccione una ubicación en su zona horaria:
Península
Ceuta y Melilla
Islas Canarias
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Particionado de discos
Este instalador puede guiarle en el particionado del disco (utilizando distintos esquemas estándar) o, si lo desea, puede hacerlo de forma manual. Si escoge el sistema de particionado guiado tendrá la oportunidad más adelante de revisar y adaptar los resultados.
Se le preguntará qué disco a utilizar si elige particionado guiado para un disco completo.
Método de particionado:
Guiado - utilizar todo el disco
Guiado - utilizar el disco completo y configurar LVM
Guiado - utilizar todo el disco y configurar LVM cifrado
Manual
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

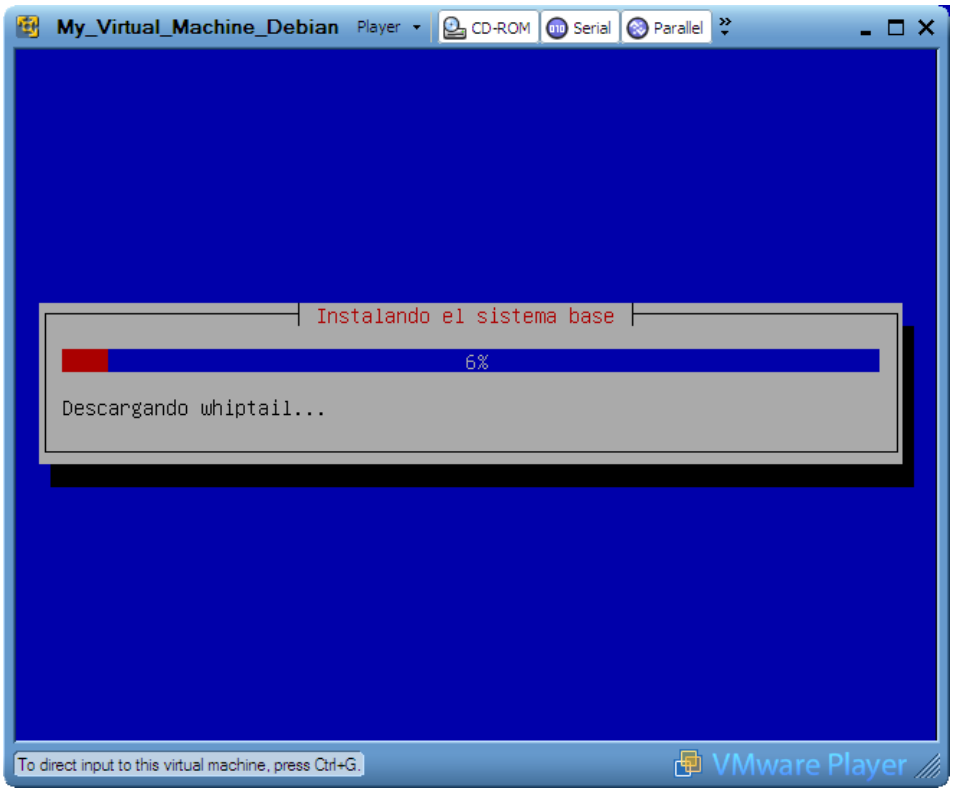

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Particionado de discos
Tenga en cuenta que se borrarán todos los datos en el disco que ha seleccionado. Este borrado no se realizará hasta que confirme que realmente quiere hacer los cambios.
Elija disco a particionar:
Maestro IDE1 (hda) - 5.0 GB VMware Virtual IDE Hard Drive
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

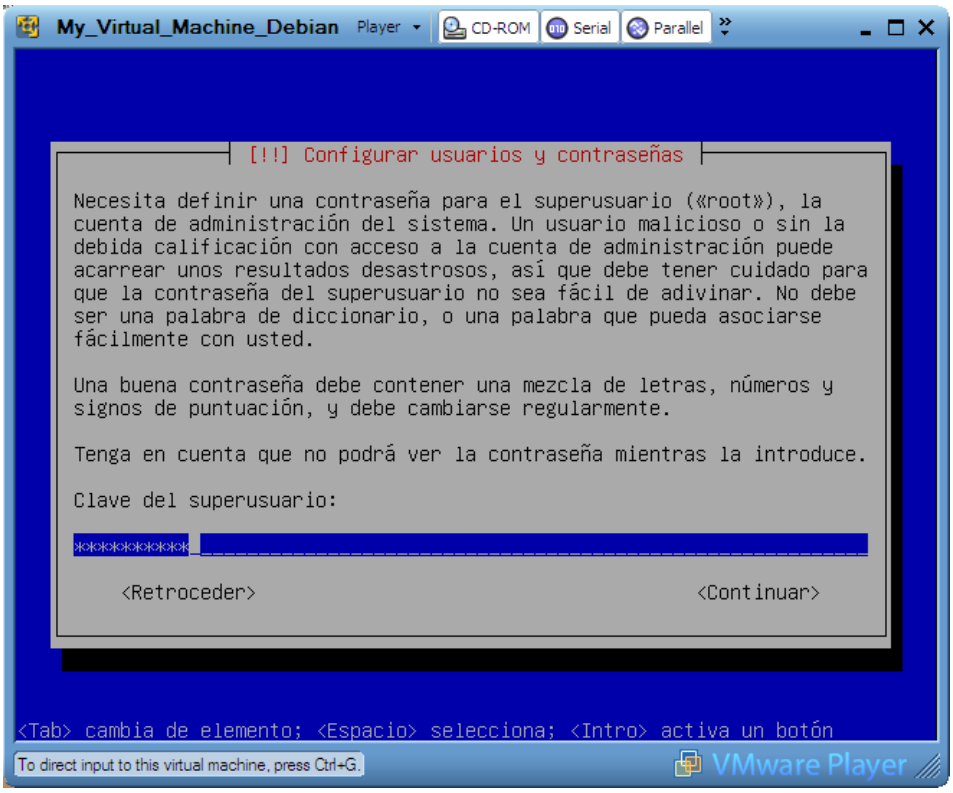

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Particionado de discos
Seleccionado para particionar:
Maestro IDE1 (hda) - VMware Virtual IDE Hard Drive: 5.0 GB
Este disco puede particionarse siguiendo uno o varios de los diferentes esquemas disponibles. Si no está seguro, escoja el primero de ellos.
Esquema de particionado:
Todos los ficheros en una partición (recomendado para novatos)
Separar la partición /home
Separar particiones /home, /usr/, /var y /tmp
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

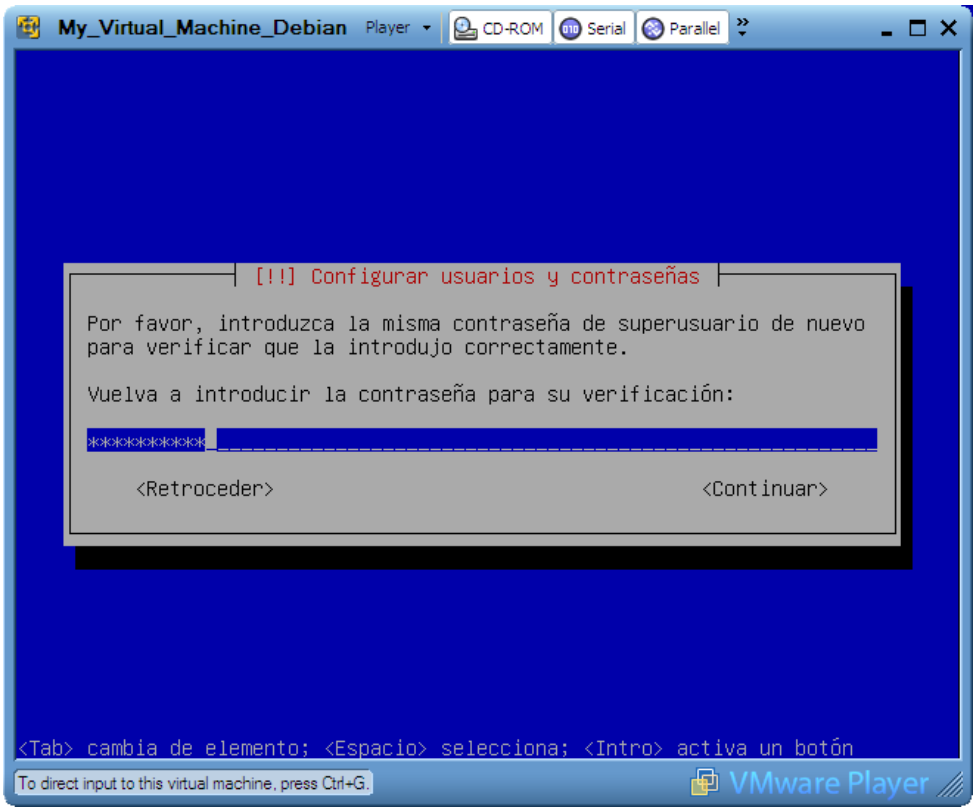

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Particionado de discos
Éste es un resumen de las particiones y puntos de montaje que tiene configurados actualmente. Seleccione una partición para modificar sus valores (sistema de ficheros, puntos de montaje, etc.), el espacio libre para añadir una partición nueva o un dispositivo para inicializar la tabla de particiones.
Particionado guiado
Ayuda del particionado
Maestro IDE1 (hda) - 5.0 GB VMware Virtual IDE Hard Drive
#1 primaria 4.8 GB B f ext3 /
#5 lógica 271.4 MB f intercambio intercambio
Deshacer los cambios realizados a las particiones
Finalizar el particionado y escribir los cambios en el disco
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

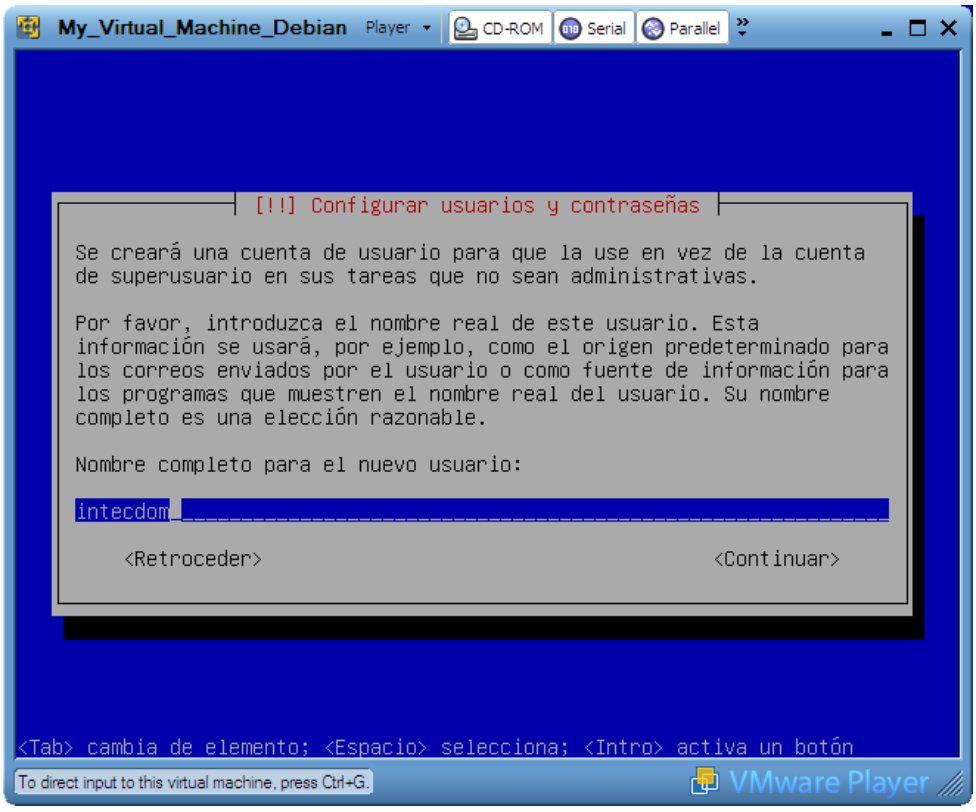

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Particionado de discos
Se escribirán en los discos todos los cambios indicados a continuación si continúa. Si no lo hace podrá hacer cambios manualmente.
AVISO: Esta operación destruirá todos los datos que existan en las particiones que haya eliminado así como en aquellas particiones que se vayan a formatear.
Se han modificado las tablas de particiones de los siguientes dispositivos:
Maestro IDE1 (hda)
Se formatearán las siguientes particiones:
partición #1 de Maestro IDE1 (hda) como ext3
partición #5 de Maestro IDE1 (hda) como intercambio
¿Desea escribir los cambios en los discos?
<Retroceder>
<Sí>
<No>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

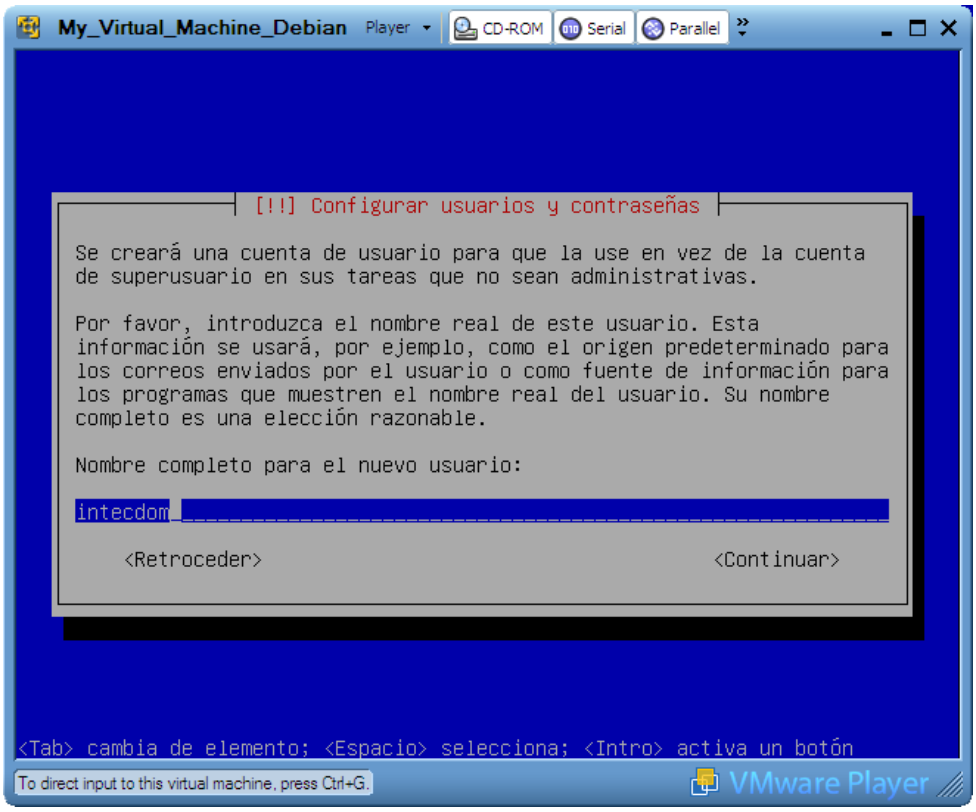
Ahora empieza la instalación de Debian

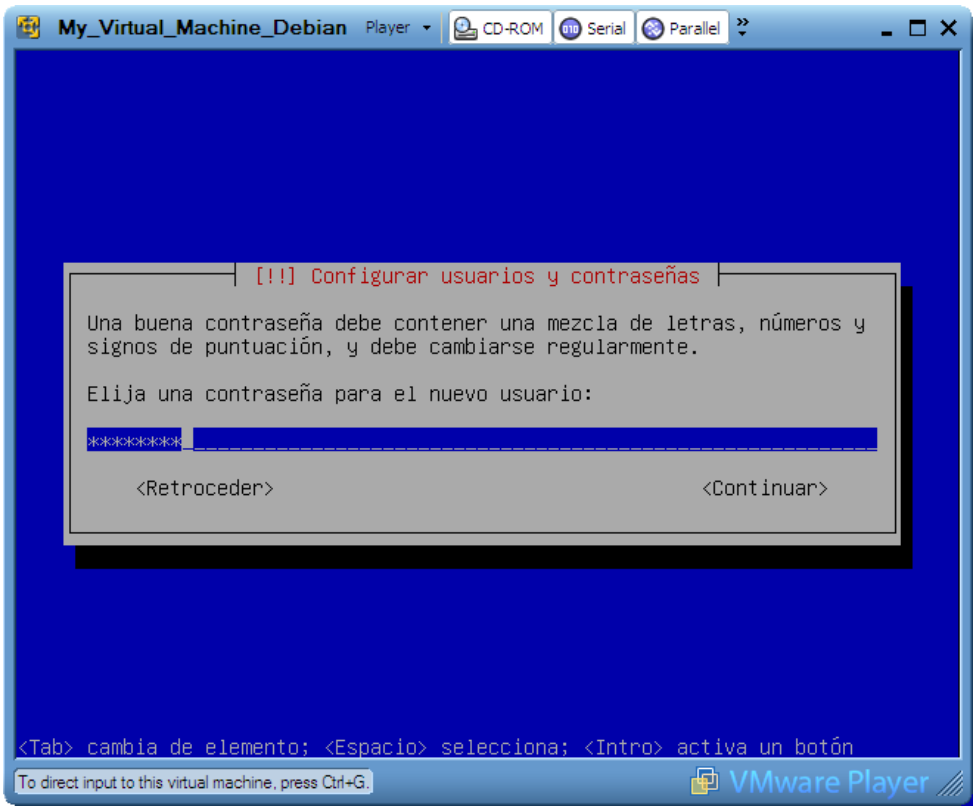


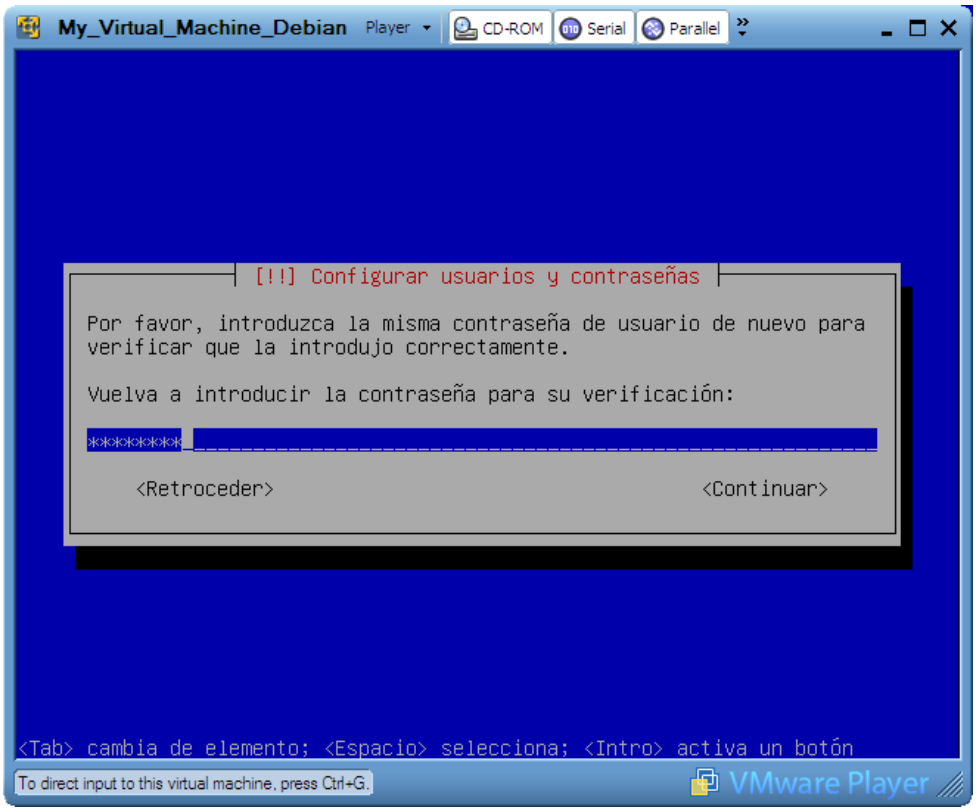


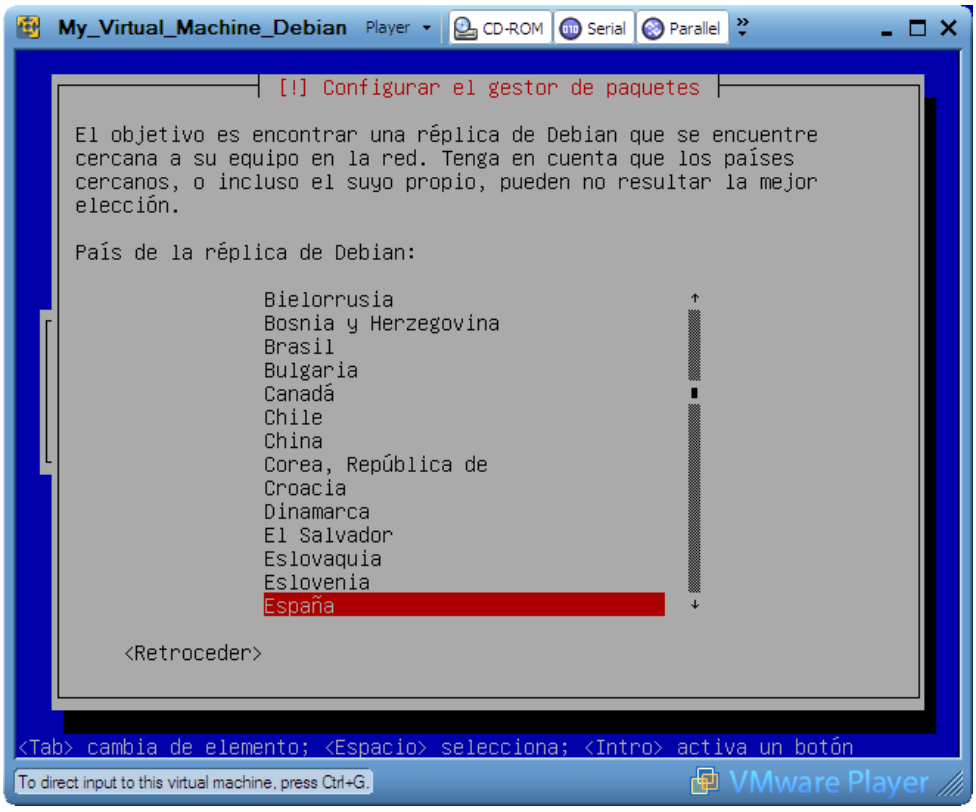

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Configurar usuarios y contraseñas
Por favor, introduzca la misma contraseña de superusuario de nuevo para verificar que la introdujo correctamente.
Vuelva a introducir la contraseña para su verificación:
****
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

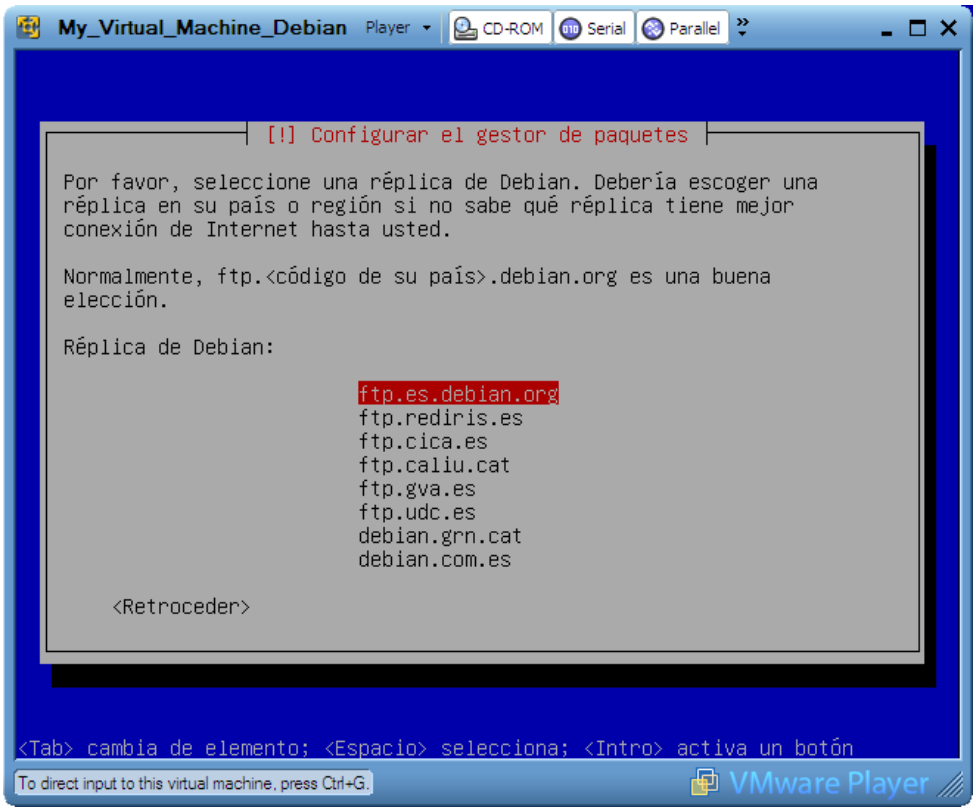

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Configurar usuarios y contraseñas
Se creará una cuenta de usuario para que la use en vez de la cuenta de superusuario en sus tareas que no sean administrativas.
Por favor, introduzca el nombre real de este usuario. Esta información se usará, por ejemplo, como el origen predeterminado para los correos enviados por el usuario o como fuente de información para los programas que muestren el nombre real del usuario. Su nombre completo es una elección razonable.
Nombre completo para el nuevo usuario:
intecdom
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

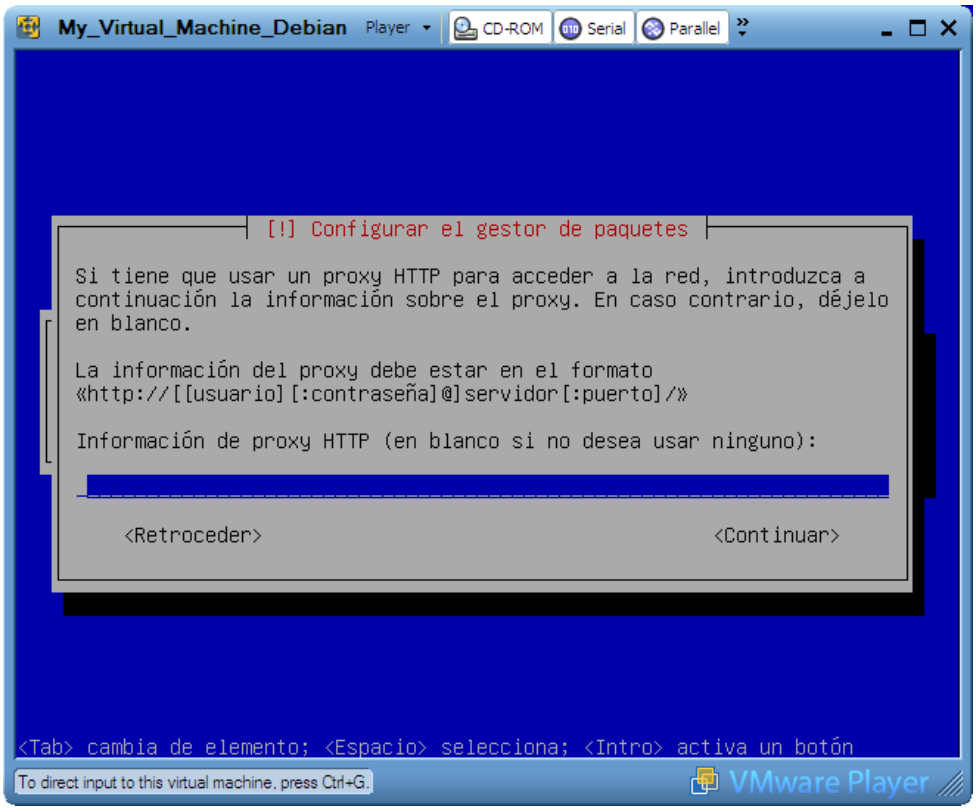

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Configurar usuarios y contraseñas
Se creará una cuenta de usuario para que la use en vez de la cuenta de superusuario en sus tareas que no sean administrativas.
Por favor, introduzca el nombre real de este usuario. Esta información se usará, por ejemplo, como el origen predeterminado para los correos enviados por el usuario o como fuente de información para los programas que muestren el nombre real del usuario. Su nombre completo es una elección razonable.
Nombre completo para el nuevo usuario:
intecdom
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

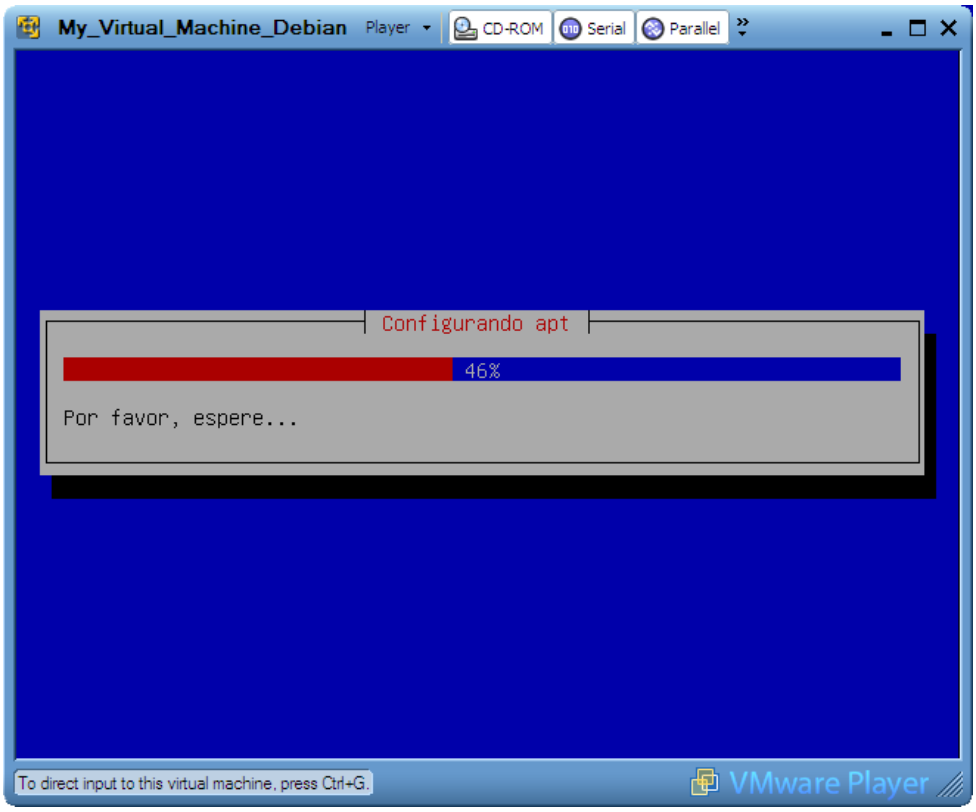

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Configurar usuarios y contraseñas
Una buena contraseña debe contener una mezcla de letras, números y signos de puntuación, y debe cambiarse regularmente.
Elija una contraseña para el nuevo usuario:
****
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

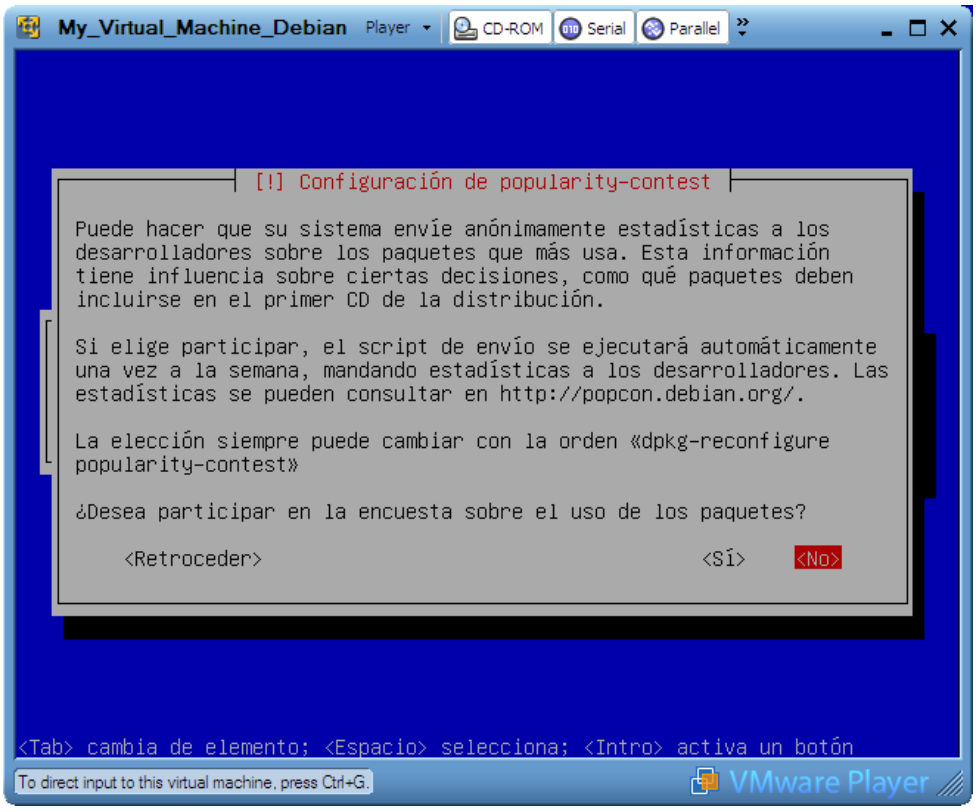

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Configurar usuarios y contraseñas
Por favor, introduzca la misma contraseña de usuario de nuevo para verificar que la introdujo correctamente.
Vuelva a introducir la contraseña para su verificación:
****
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar el gestor de paquetes
El objetivo es encontrar una réplica de Debian que se encuentre cercana a su equipo en la red. Tenga en cuenta que los países cercanos, o incluso el suyo propio, pueden no resultar la mejor elección.
País de la réplica de Debian:
Bielorrusia
Bosnia y Herzegovina
Brasil
Bulgaria
Canadá
Chile
China
Corea, República de
Croacia
Dinamarca
El Salvador
Eslovaquia
Eslovenia
España
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

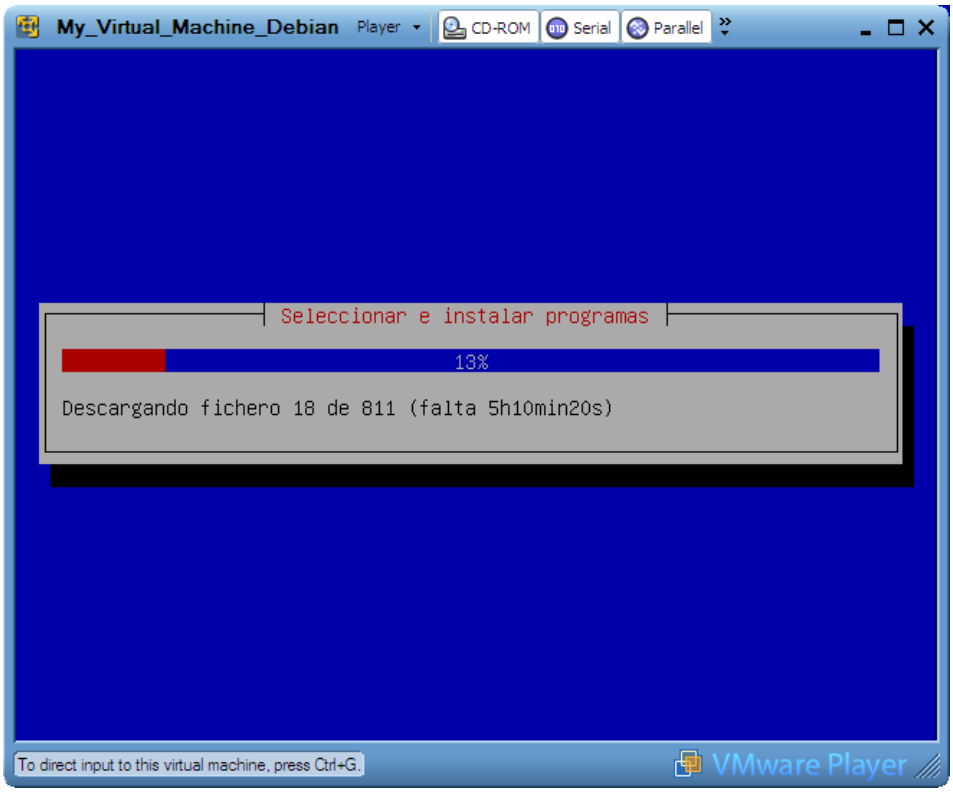

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar el gestor de paquetes
Por favor, seleccione una réplica de Debian. Debería escoger una réplica en su país o región si no sabe qué réplica tiene mejor conexión de Internet hasta usted.
Normalmente, ftp.<código de su país>.debian.org es una buena elección.
Réplica de Debian:
ftp.es.debian.org
ftp.rediris.es
ftp.cica.es
ftp.caliu.cat
ftp.gva.es
ftp.udc.es
debian.grn.cat
debian.com.es
<Retroceder>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

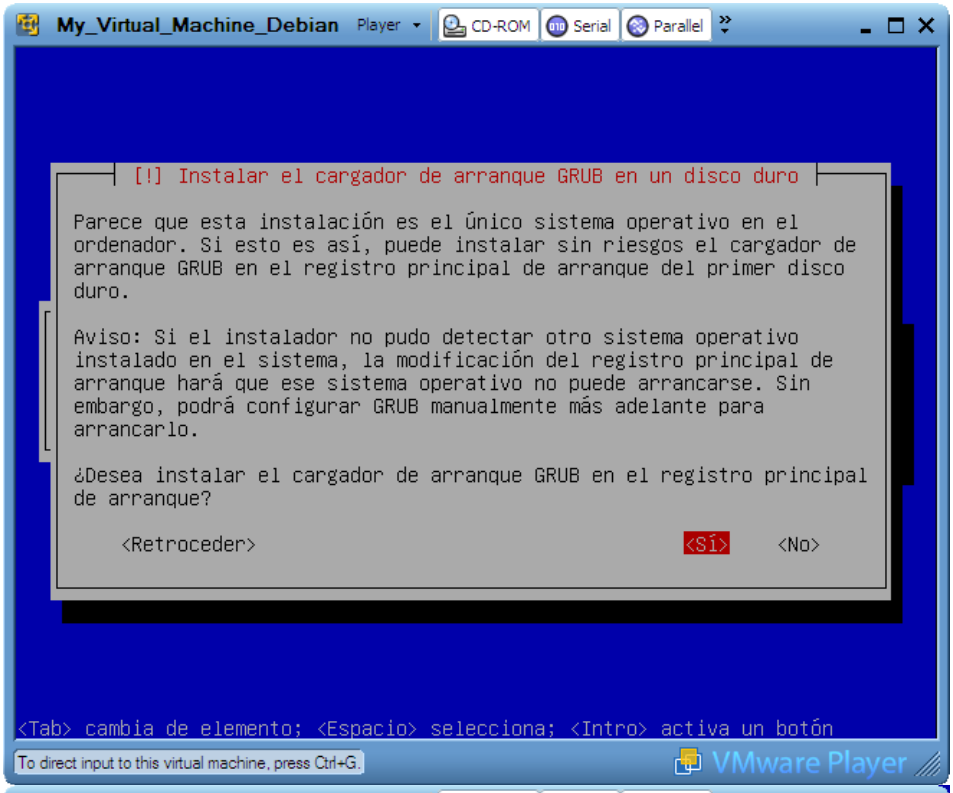

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configurar el gestor de paquetes
Si tiene que usar un proxy HTTP para acceder a la red, introduzca a continuación la información sobre el proxy. En caso contrario, déjelo en blanco.
La información del proxy debe estar en el formato «http://[[usuario][:contraseña]@]servidor[:puerto]/»
Información de proxy HTTP (en blanco si no desea usar ninguno):
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

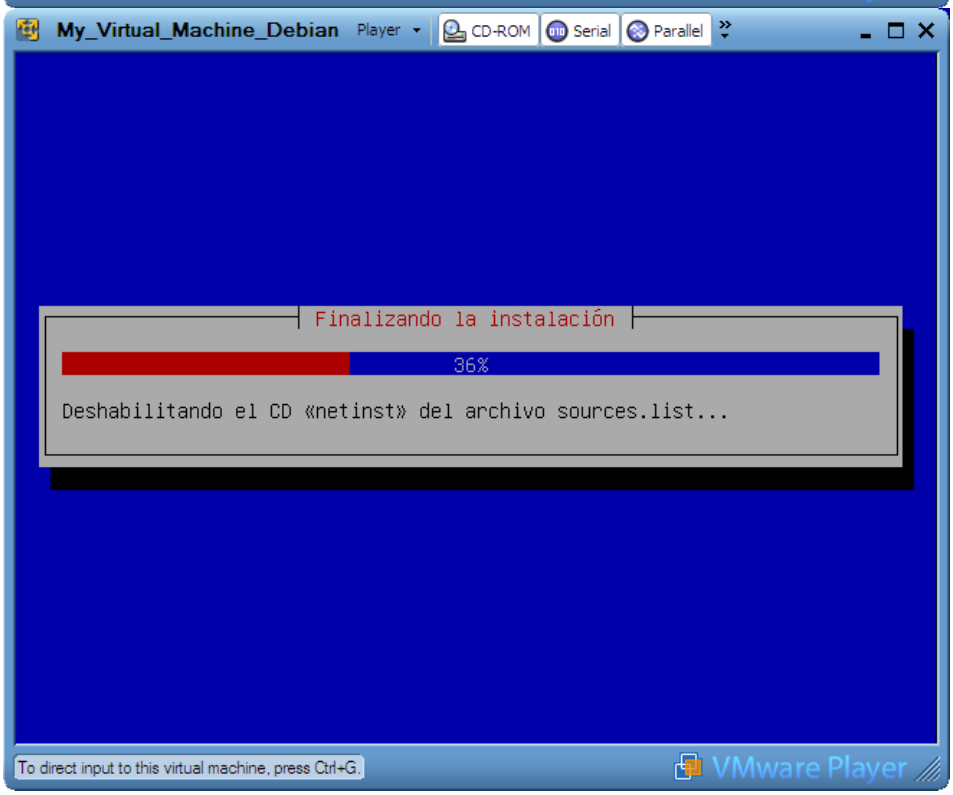

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Configurando apt
46%
Por favor, espere...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

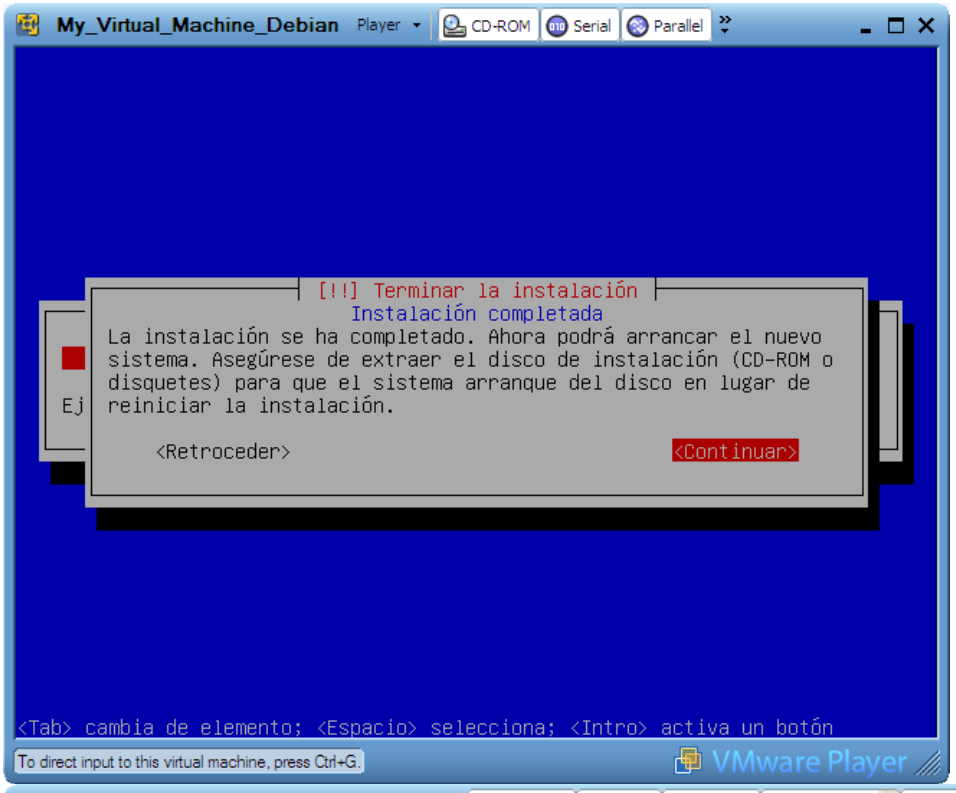

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Configuración de popularity-contest
Puede hacer que su sistema envíe anónimamente estadísticas a los desarrolladores sobre los paquetes que más usa. Esta información tiene influencia sobre ciertas decisiones, como qué paquetes deben incluirse en el primer CD de la distribución.
Si elige participar, el script de envío se ejecutará automáticamente una vez a la semana, mandando estadísticas a los desarrolladores. Las estadísticas se pueden consultar en http://popcon.debian.org/.
La elección siempre puede cambiar con la orden «dpkg-reconfigure popularity-contest»
¿Desea participar en la encuesta sobre el uso de los paquetes?
<Retroceder>
<Sí>
<No>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

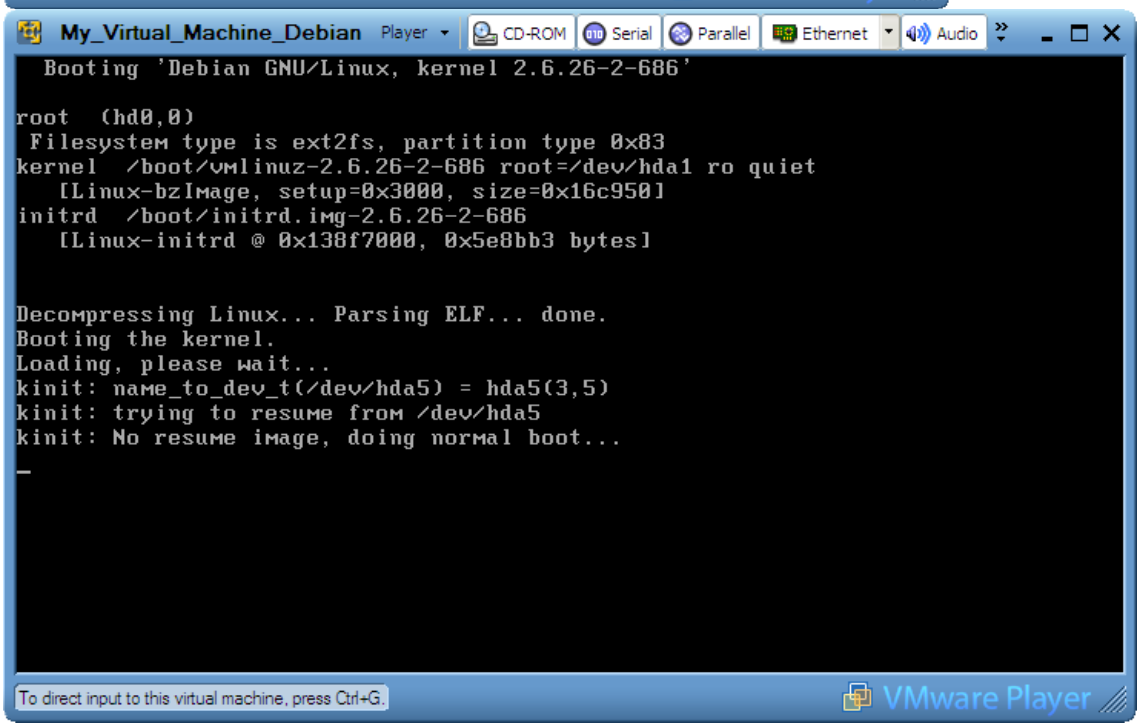
A continuación habría una pantalla adicional, en la que decimos si queremos instalar sólo el entorno de escritorio o más cosas, como servidor Web, servidor de correo, etc, y tras ella comienza la instalación:




My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!] Instalar el cargador de arranque GRUB en un disco duro
Parece que esta instalación es el único sistema operativo en el ordenador. Si esto es así, puede instalar sin riesgos el cargador de arranque GRUB en el registro principal de arranque del primer disco duro.
Aviso: Si el instalador no pudo detectar otro sistema operativo instalado en el sistema, la modificación del registro principal de arranque hará que ese sistema operativo no puede arrancarse. Sin embargo, podrá configurar GRUB manualmente más adelante para arrancarlo.
¿Desea instalar el cargador de arranque GRUB en el registro principal de arranque?
<Retroceder>
<Sí>
<No>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Finalizando la instalación
36%
Deshabilitando el CD «netinst» del archivo sources.list...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

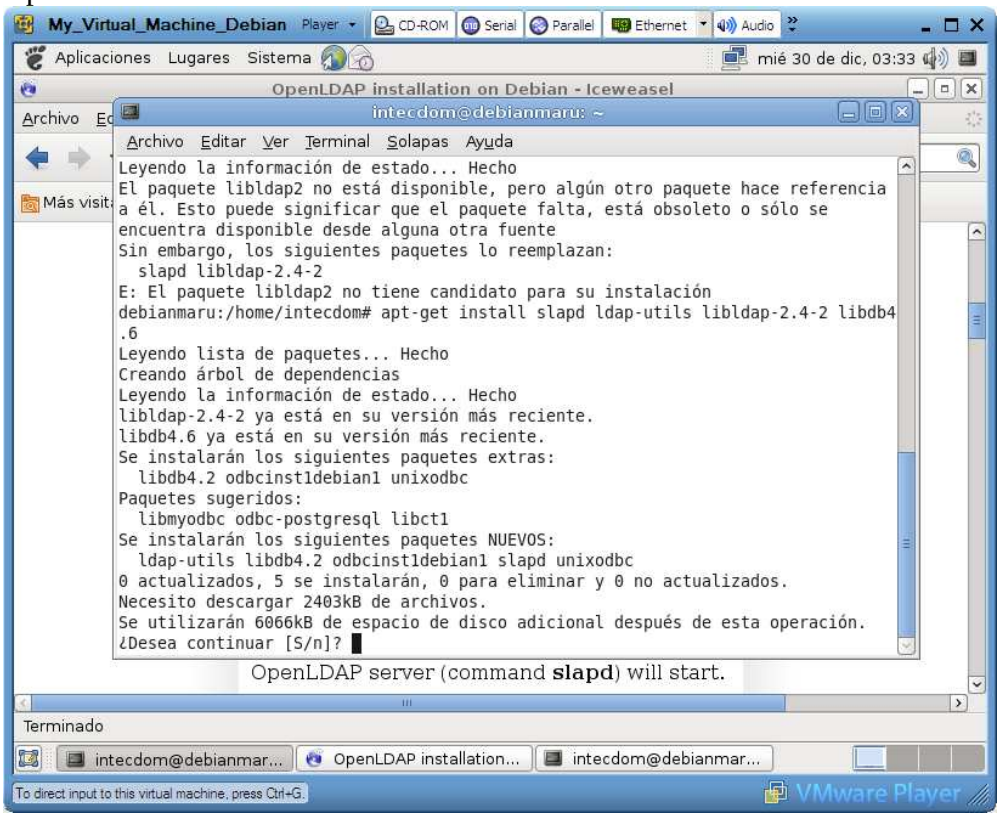

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
[!!] Terminar la instalación
Instalación completada
La instalación se ha completado. Ahora podrá arrancar el nuevo sistema. Asegúrese de extraer el disco de instalación (CD-ROM o disquetes) para que el sistema arranque del disco en lugar de reiniciar la instalación.
<Retroceder>
<Continuar>
<Tab> cambia de elemento; <Espacio> selecciona; <Intro> activa un botón
To direct input to this virtual machine, press Ctrl+G.
VMware Player

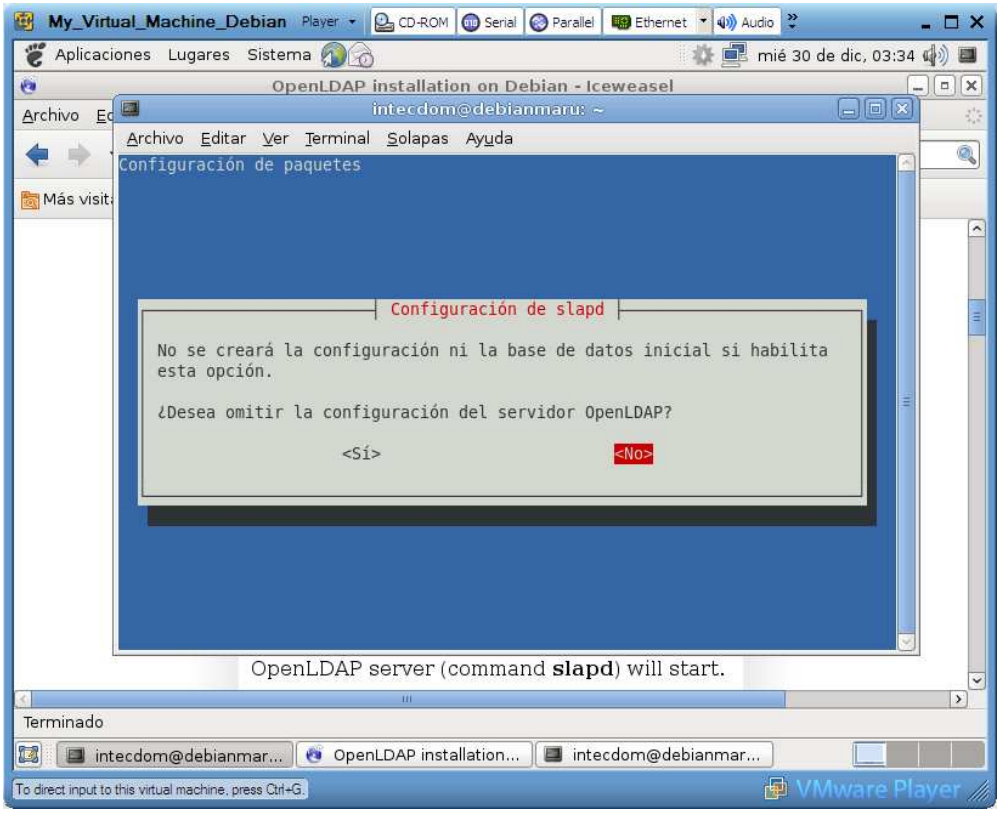

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Booting 'Debian GNU/Linux, kernel 2.6.26-2-686'
root (hd0,0)
Filesystem type is ext2fs, partition type 0x83
kernel /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro quiet
[Linux-bzImage, setup=0x3000, size=0x16c950]
initrd /boot/initrd.img-2.6.26-2-686
[Linux-initrd @ 0x138f7000, 0x5e8bb3 bytes]
Decompressing Linux... Parsing ELF... done.
Booting the kernel.
Loading, please wait...
kinit: name_to_dev_t(/dev/hda5) = hda5(3,5)
kinit: trying to resume from /dev/hda5
kinit: No resume image, doing normal boot...
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
debian
Bienvenido
Usuario:
Introduzca su nombre de usuario
intecdom
2 Microsoft Office ...
¿Como hago para ba...
links_manuales_proy...
2 Windows Explorer
My_Virtual_Machine_...

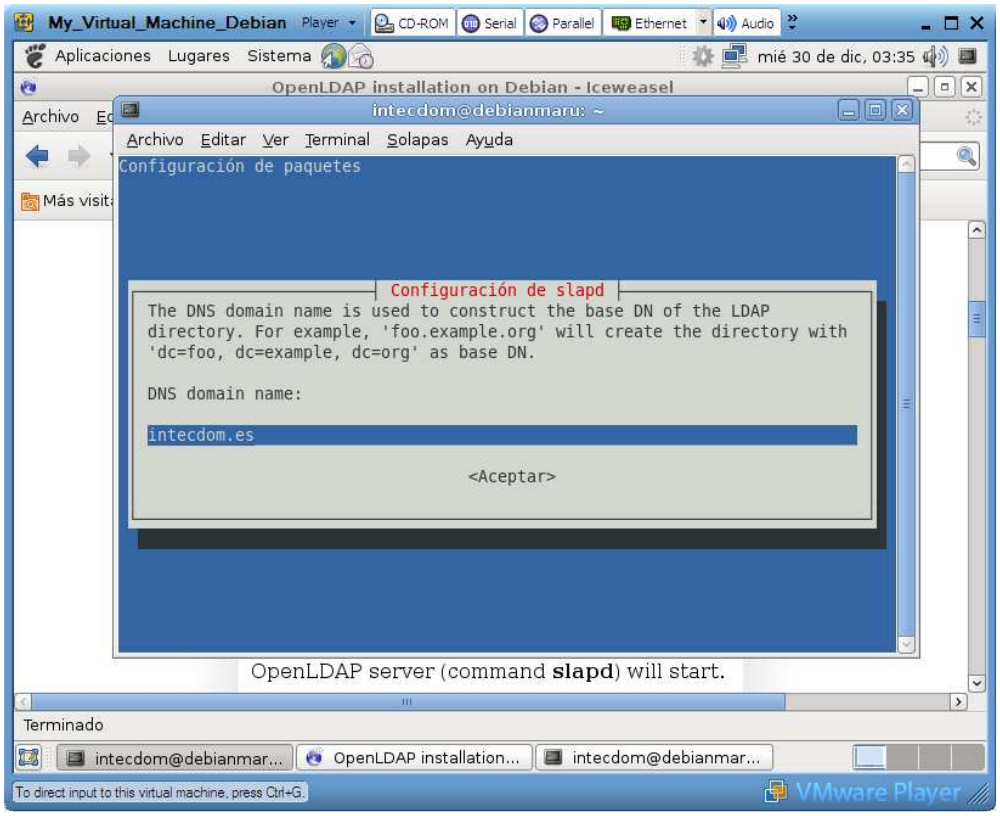

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
debian
Bienvenido
Contraseña:
2 Microsoft Office ...
¿Como hago para ba...
links_manuales_proy...
2 Windows Explorer
My_Virtual_Machine_...

Primero hacemos:
Aptitude install LDAP

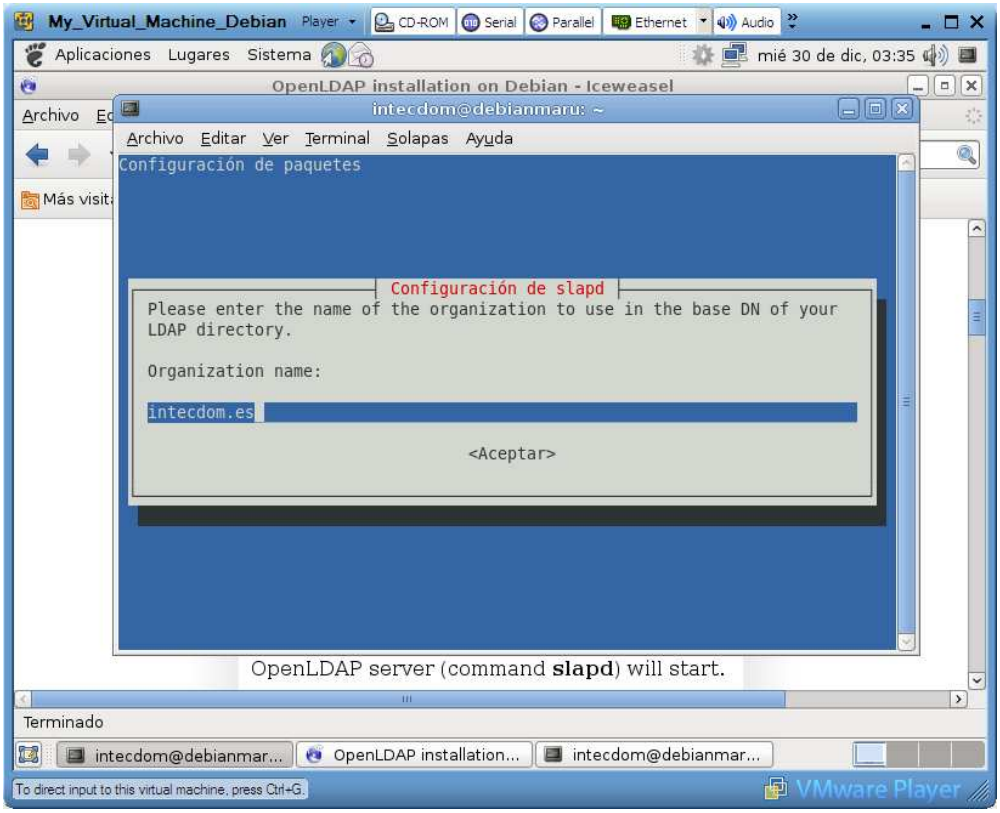




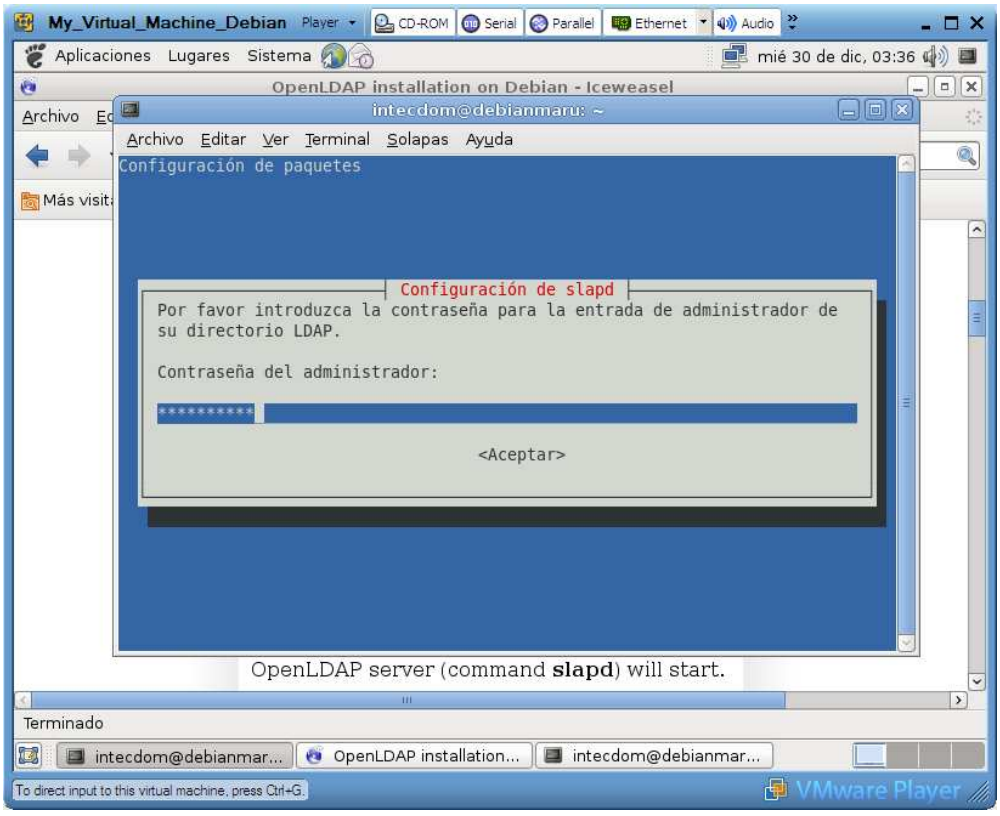
OpenLDAP server (command **slapd**) will start.



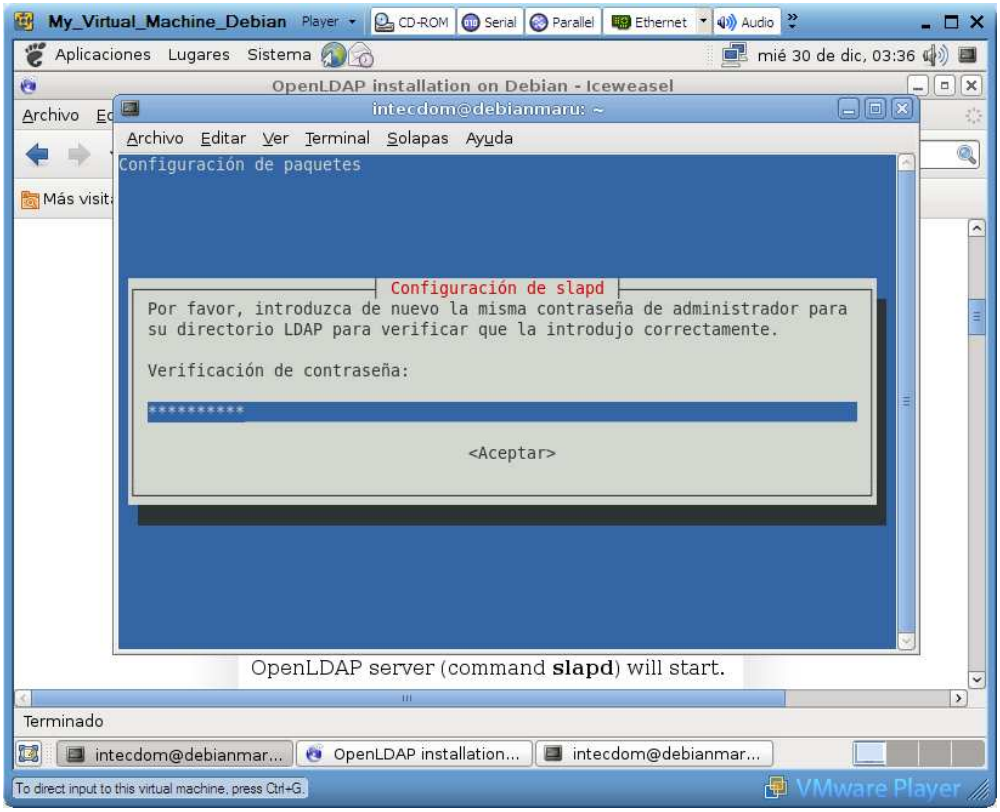
OpenLDAP server (command **slapd**) will start.



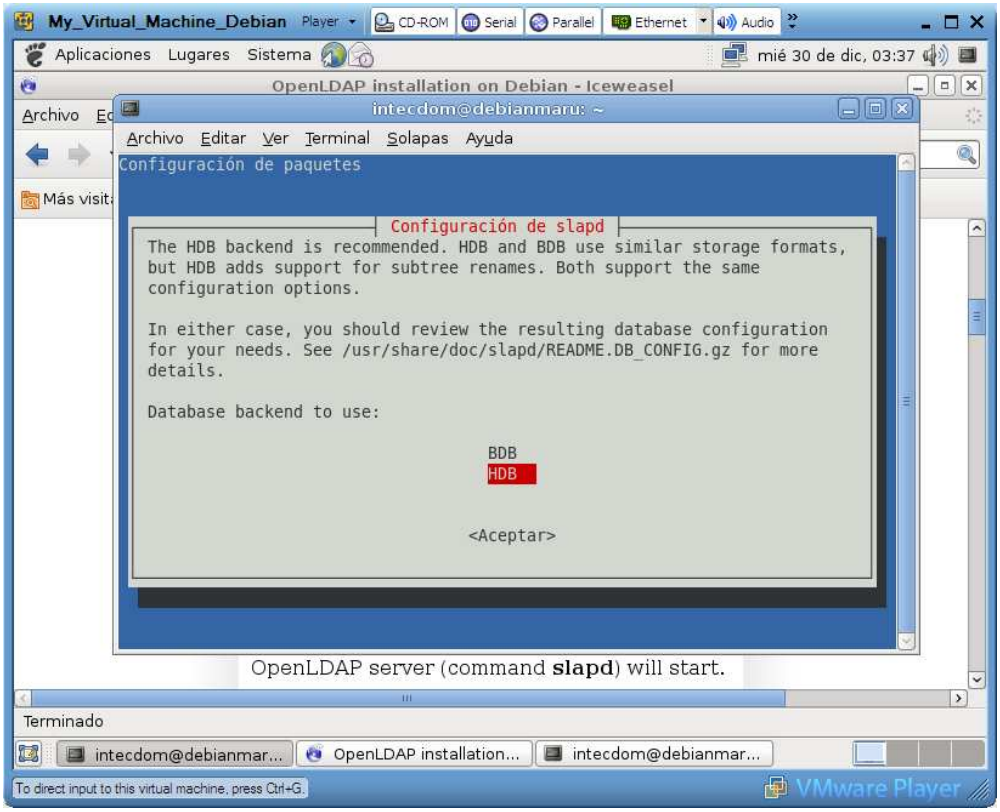
OpenLDAP server (command **slapd**) will start.



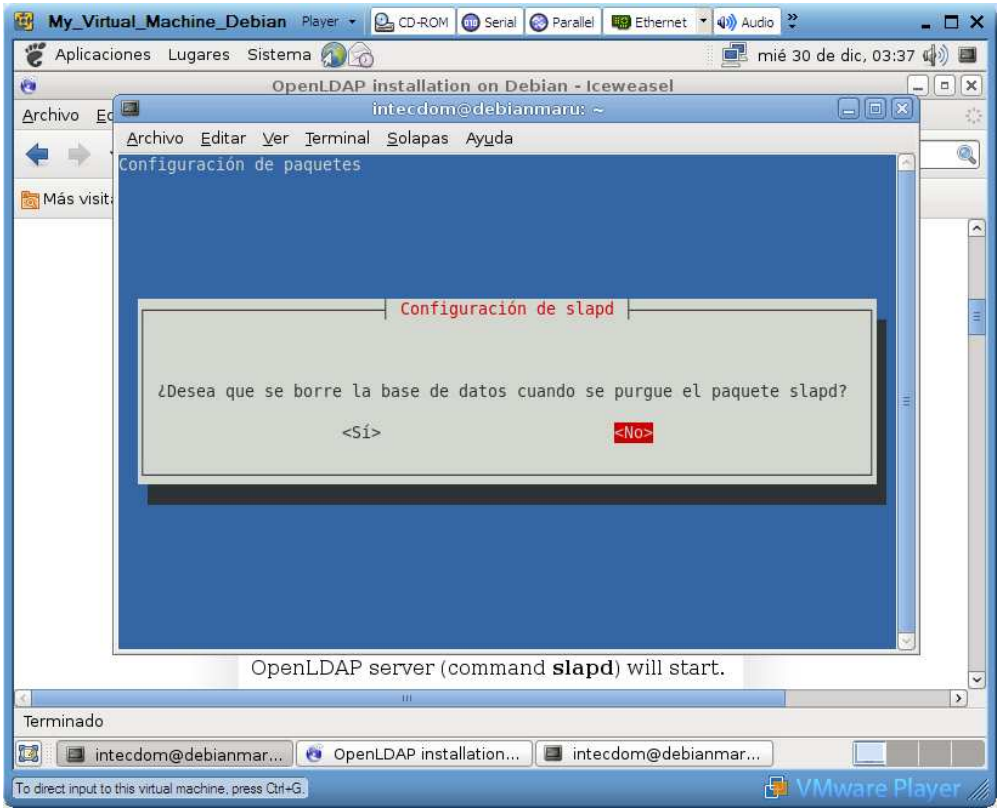
OpenLDAP server (command **slapd**) will start.

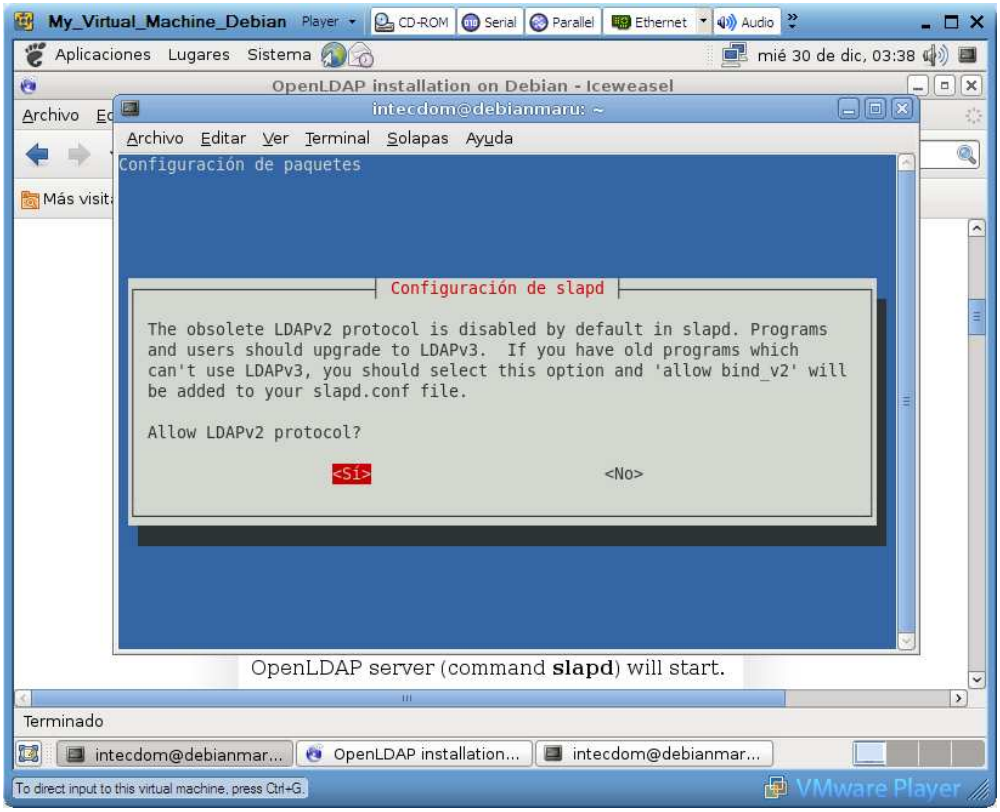


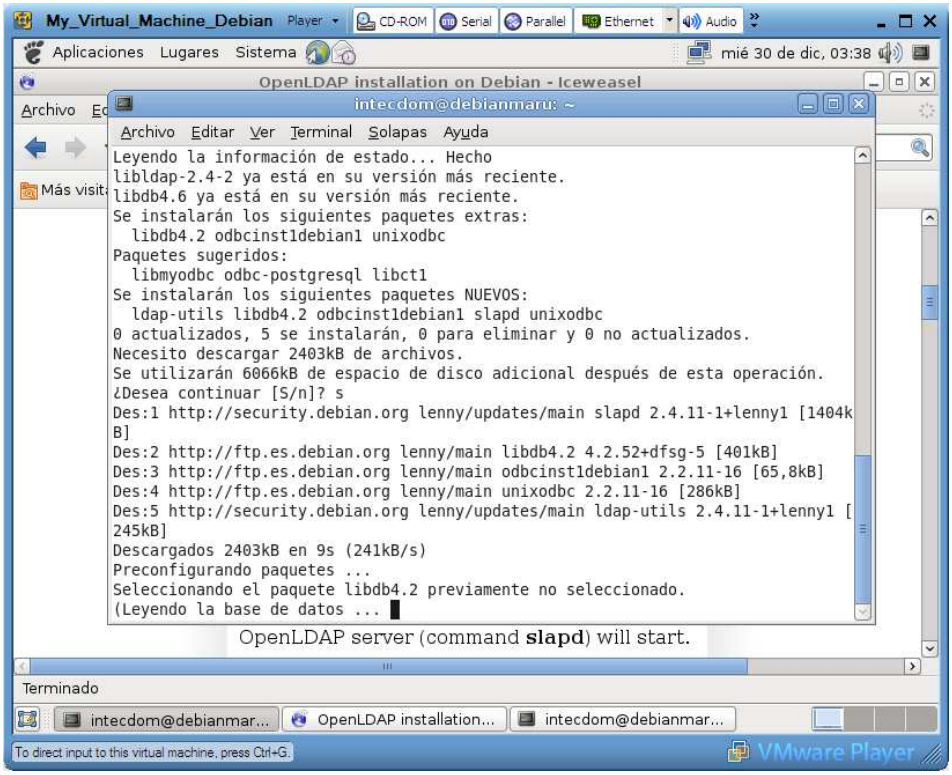
OpenLDAP server (command **slapd**) will start.



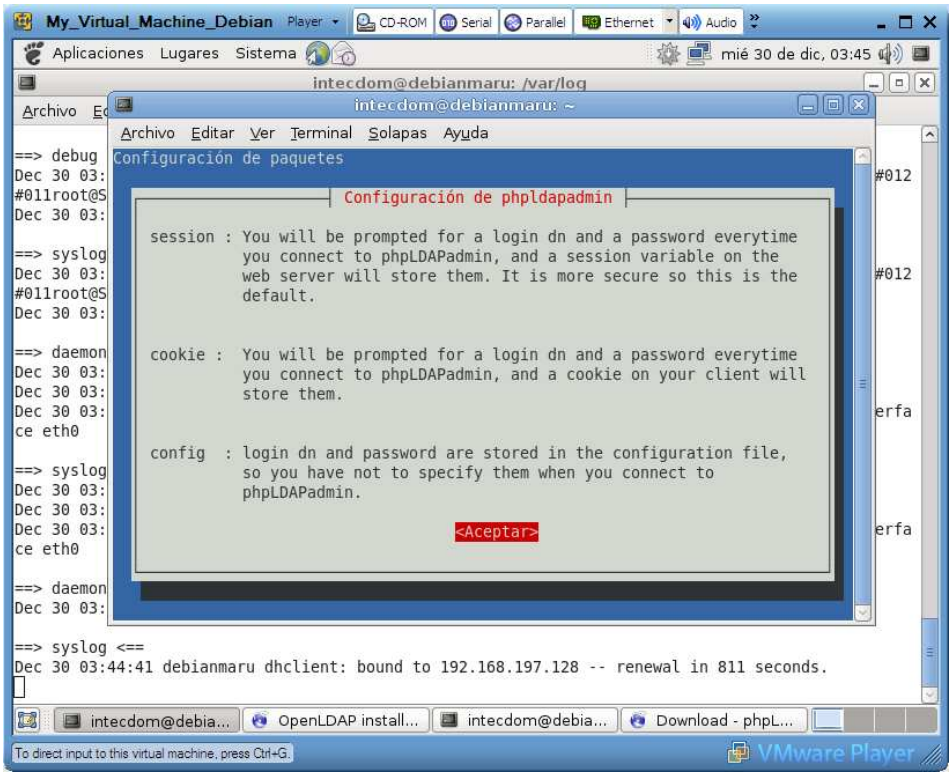
OpenLDAP server (command **slapd**) will start.

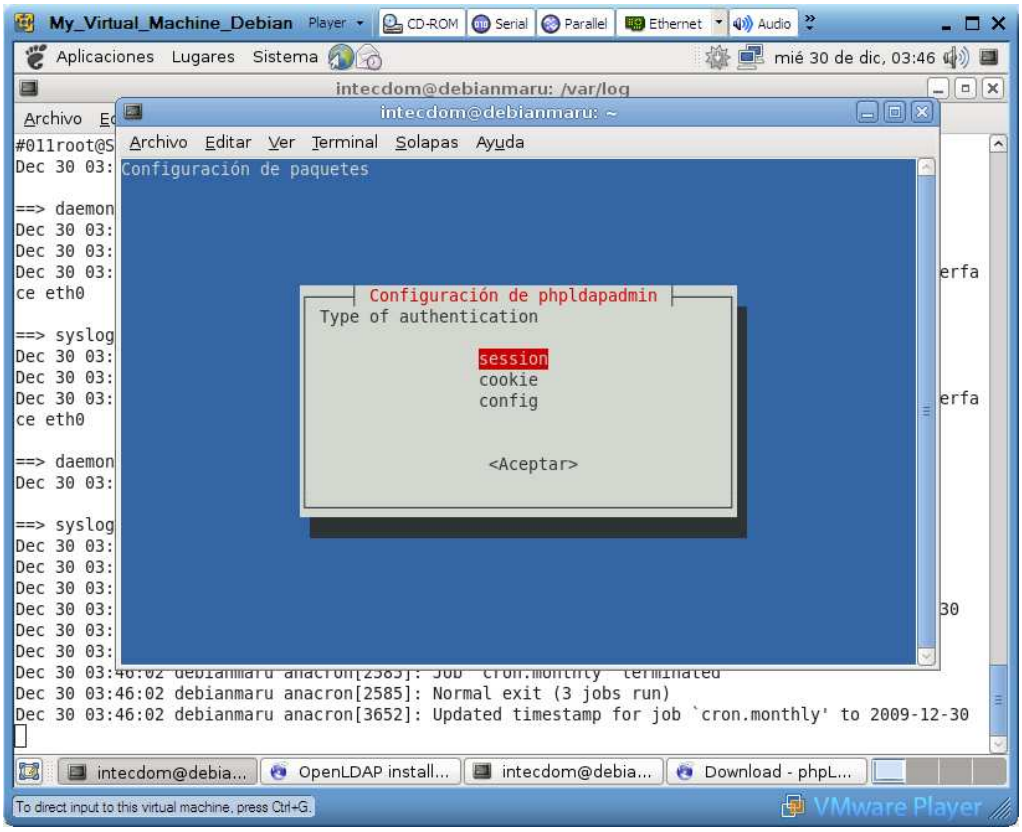

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:37
OpenLDAP installation on Debian - Iceweasel
Archivo
intecdom@debianmaru: ~
Archivo Editar Ver Terminal Solapas Ayuda
Configuración de paquetes
Configuración de slapd
¿Desea que se borre la base de datos cuando se purgue el paquete slapd?
<Sí>
<No>
OpenLDAP server (command slapd) will start.
Terminado
intecdom@debianmar...
OpenLDAP installation...
intecdom@debianmar...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

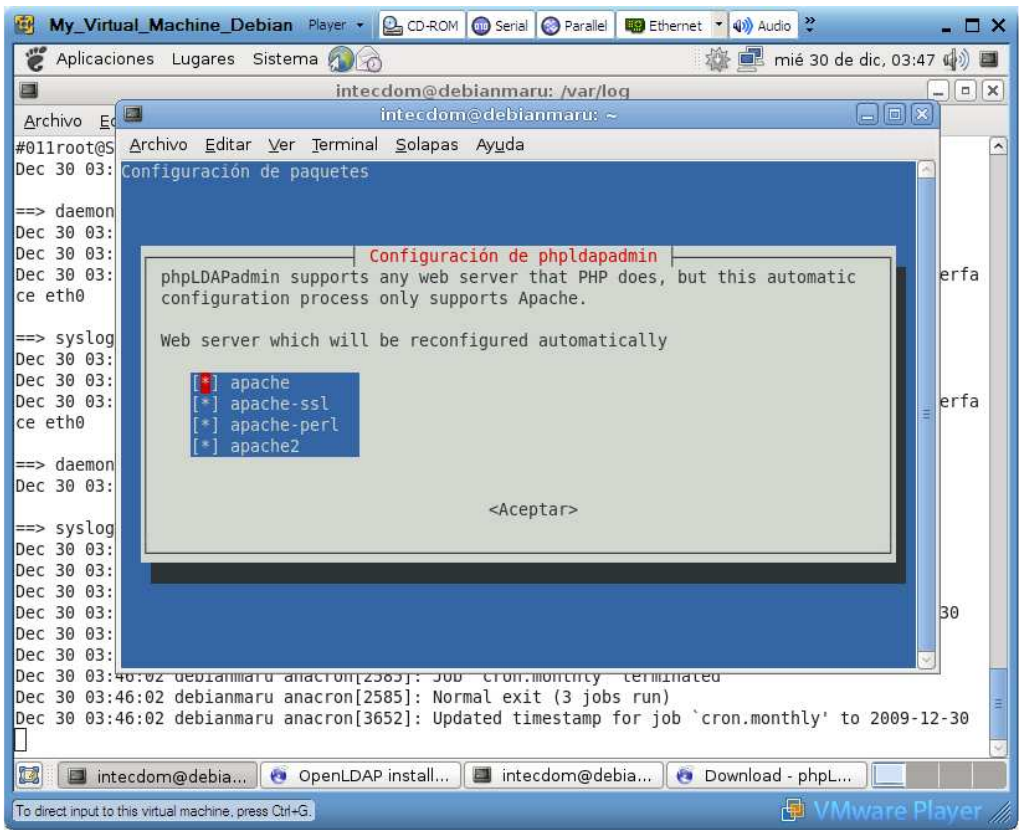

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:38
OpenLDAP installation on Debian - Iceweasel
Archivo
intecdom@debianmaru: ~
Archivo Editar Ver Terminal Solapas Ayuda
Configuración de paquetes
Configuración de slapd
The obsolete LDAPv2 protocol is disabled by default in slapd. Programs
and users should upgrade to LDAPv3. If you have old programs which
can't use LDAPv3, you should select this option and 'allow bind_v2' will
be added to your slapd.conf file.
Allow LDAPv2 protocol?
<Si>
<No>
OpenLDAP server (command slapd) will start.
Terminado
intecdom@debianmar...
OpenLDAP installation...
intecdom@debianmar...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

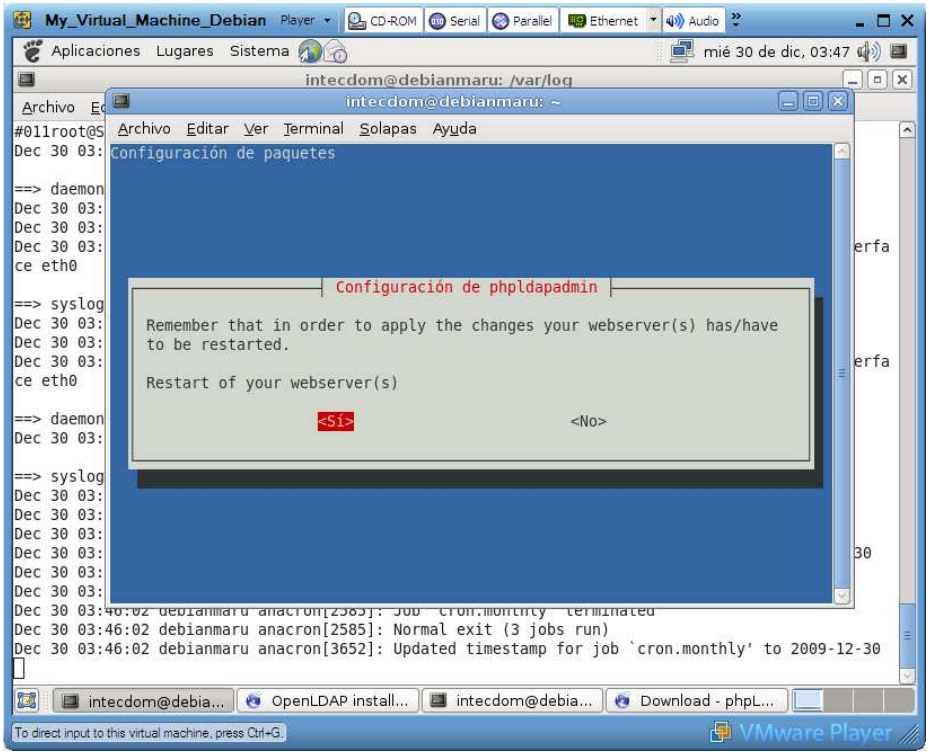


Aptitude install phpldapadmin

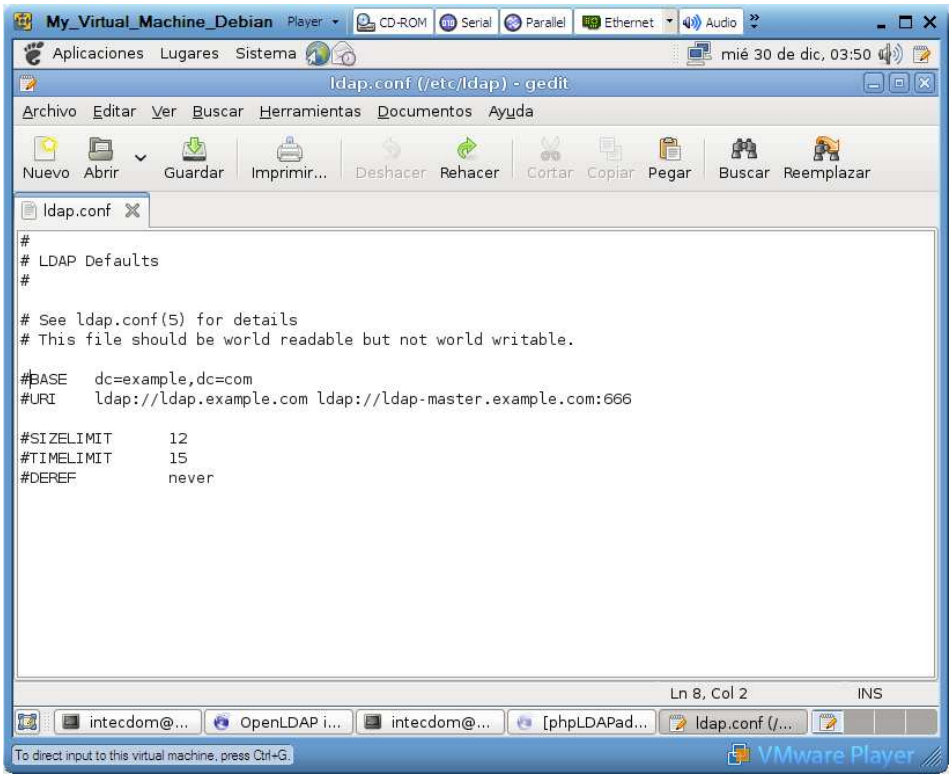


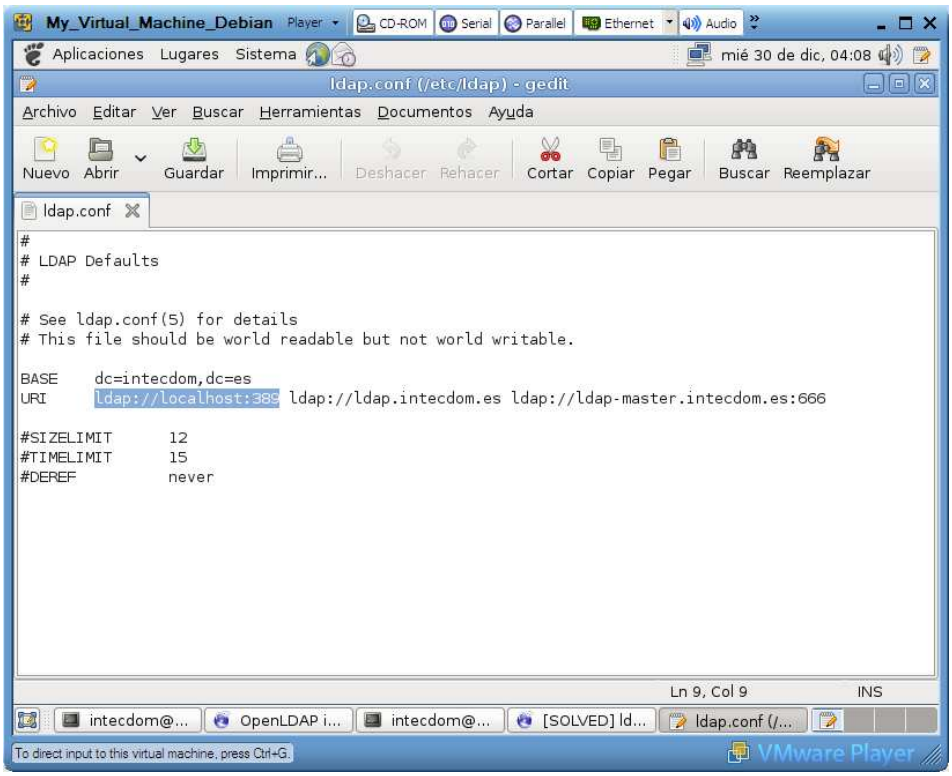

My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:46
intecdom@debianmaru: /var/log
intecdom@debianmaru: ~
Archivo
Editar
Ver
Terminal
Solapas
Ayuda
Configuración de paquetes
Configuración de phpldapadmin
Type of authentication
session
cookie
config
<Aceptar>
Dec 30 03:46:02 debianmaru anacron[2585]: Normal exit (3 jobs run)
Dec 30 03:46:02 debianmaru anacron[3652]: Updated timestamp for job `cron.monthly' to 2009-12-30
intecdom@debia...
OpenLDAP install...
intecdom@debia...
Download - phpL...
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:47
intecdom@debianmaru: /var/log
intecdom@debianmaru: ~
Archivo
Editar
Ver
Terminal
Solapas
Ayuda
Configuración de paquetes
Configuración de phpldapadmin
phpLDAPadmin supports any web server that PHP does, but this automatic
configuration process only supports Apache.
Web server which will be reconfigured automatically
[*] apache
[*] apache-ssl
[*] apache-perl
[*] apache2
<Aceptar>
Dec 30 03:46:02 debianmaru anacron[2585]: Normal exit (3 jobs run)
Dec 30 03:46:02 debianmaru anacron[3652]: Updated timestamp for job `cron.monthly' to 2009-12-30
intecdom@debia...
OpenLDAP install...
intecdom@debia...
Download - phpL...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

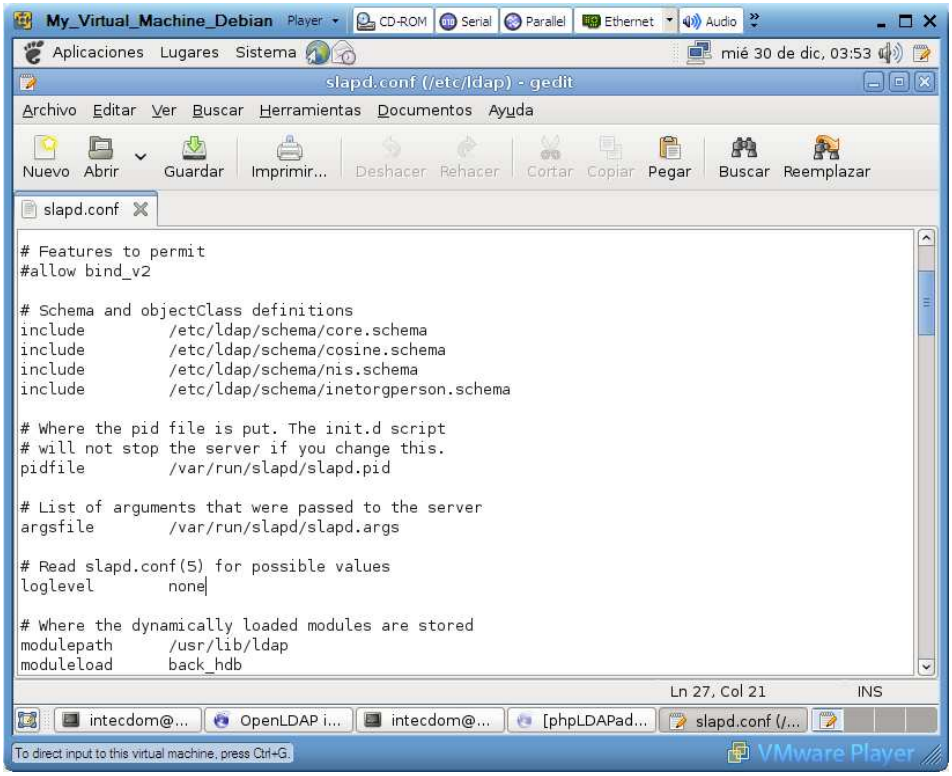


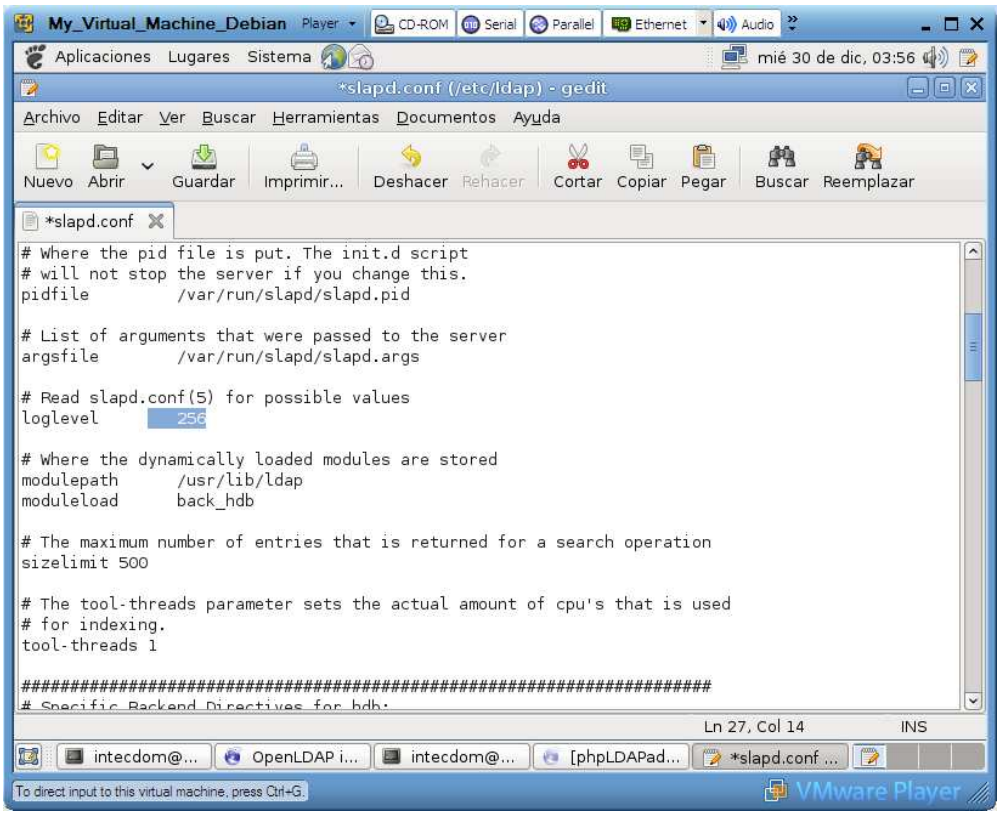
Cd /etc/ldap
Gedit ldap.conf

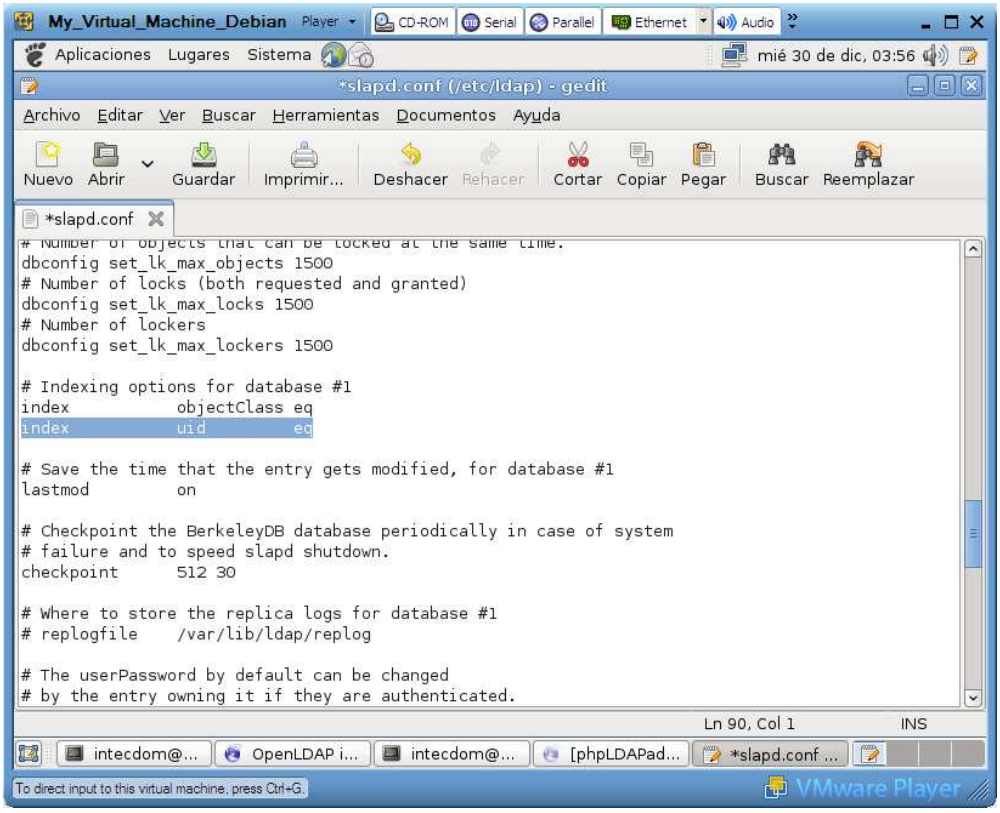




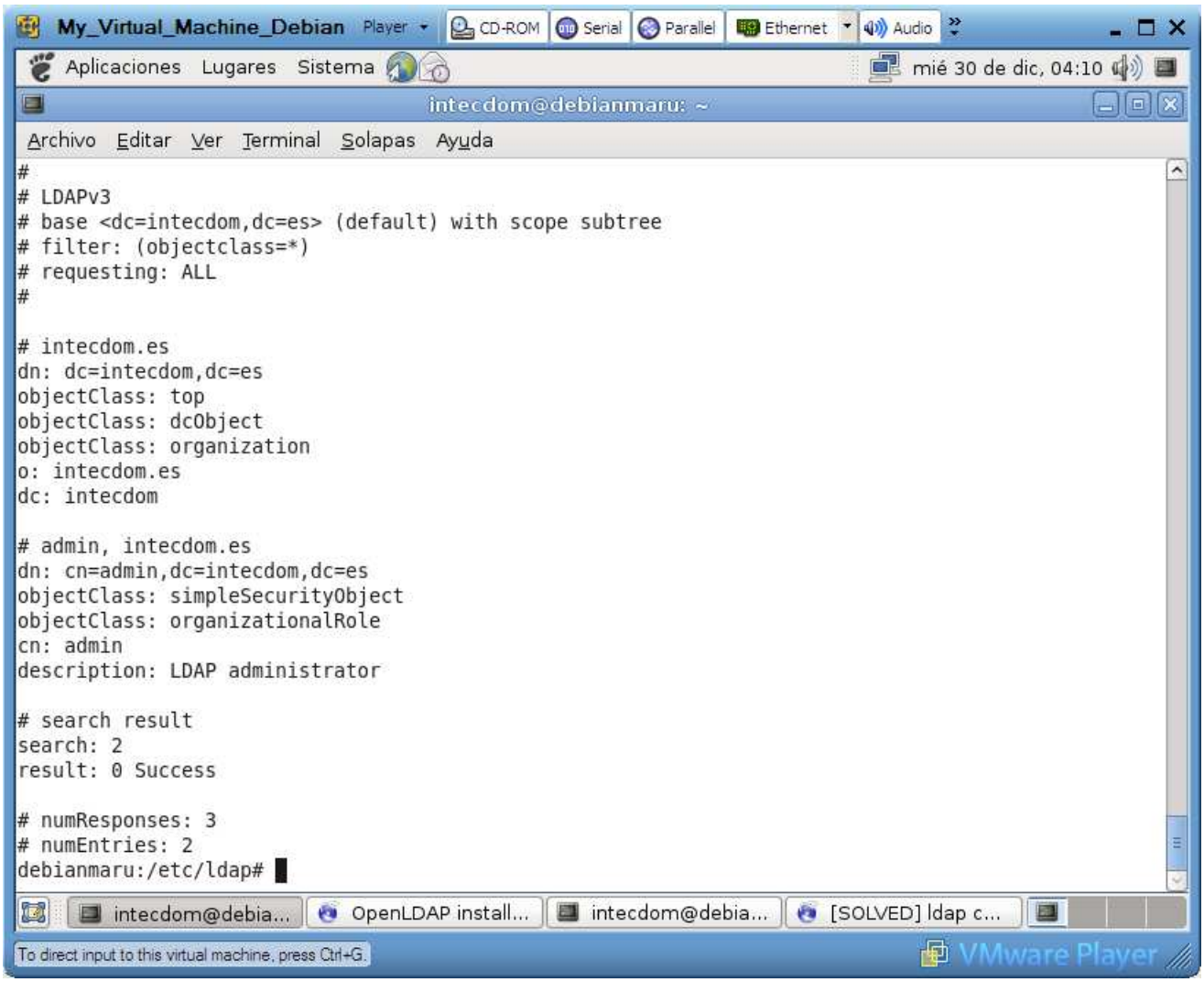
Gedit slapd.conf




My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:56
*slapd.conf (/etc/ldap) - gedit
Archivo
Editar
Ver
Buscar
Herramientas
Documentos
Ayuda
Nuevo
Abrir
Guardar
Imprimir...
Deshacer
Rehacer
Cortar
Copiar
Pegar
Buscar
Reemplazar
*slapd.conf
Where the pid file is put. The init.d script
will not stop the server if you change this.
pidfile /var/run/slapd/slapd.pid
List of arguments that were passed to the server
argsfile /var/run/slapd/slapd.args
Read slapd.conf(5) for possible values
loglevel 256
Where the dynamically loaded modules are stored
modulepath /usr/lib/ldap
moduleload back_hdb
The maximum number of entries that is returned for a search operation
sizelimit 500
The tool-threads parameter sets the actual amount of cpu's that is used
for indexing.
tool-threads 1
Ln 27, Col 14
INS
intecdom@...
OpenLDAP i...
intecdom@...
[phpLDAPad...
*slapd.conf ...
To direct input to this virtual machine, press Ctrl+G.
VMware Player


My_Virtual_Machine_Debian
Player
CD-ROM
Serial
Parallel
Ethernet
Audio
Aplicaciones
Lugares
Sistema
mié 30 de dic, 03:56
*slapd.conf (/etc/ldap) - gedit
Archivo
Editar
Ver
Buscar
Herramientas
Documentos
Ayuda
Nuevo
Abrir
Guardar
Imprimir...
Deshacer
Rehacer
Cortar
Copiar
Pegar
Buscar
Reemplazar
*slapd.conf
dbconfig set_lk_max_objects 1500
Number of locks (both requested and granted)
dbconfig set_lk_max_locks 1500
Number of lockers
dbconfig set_lk_max_lockers 1500
Indexing options for database #1
index objectClass eq
index uid eq
Save the time that the entry gets modified, for database #1
lastmod on
Checkpoint the BerkeleyDB database periodically in case of system
failure and to speed slapd shutdown.
checkpoint 512 30
Where to store the replica logs for database #1
replogfile /var/lib/ldap/replog
The userPassword by default can be changed
by the entry owning it if they are authenticated.
Ln 90, Col 1
INS
intecdom@...
OpenLDAP i...
intecdom@...
[phpLDAPad...
*slapd.conf ...
To direct input to this virtual machine, press Ctrl+G.
VMware Player

Ldapsearch –x

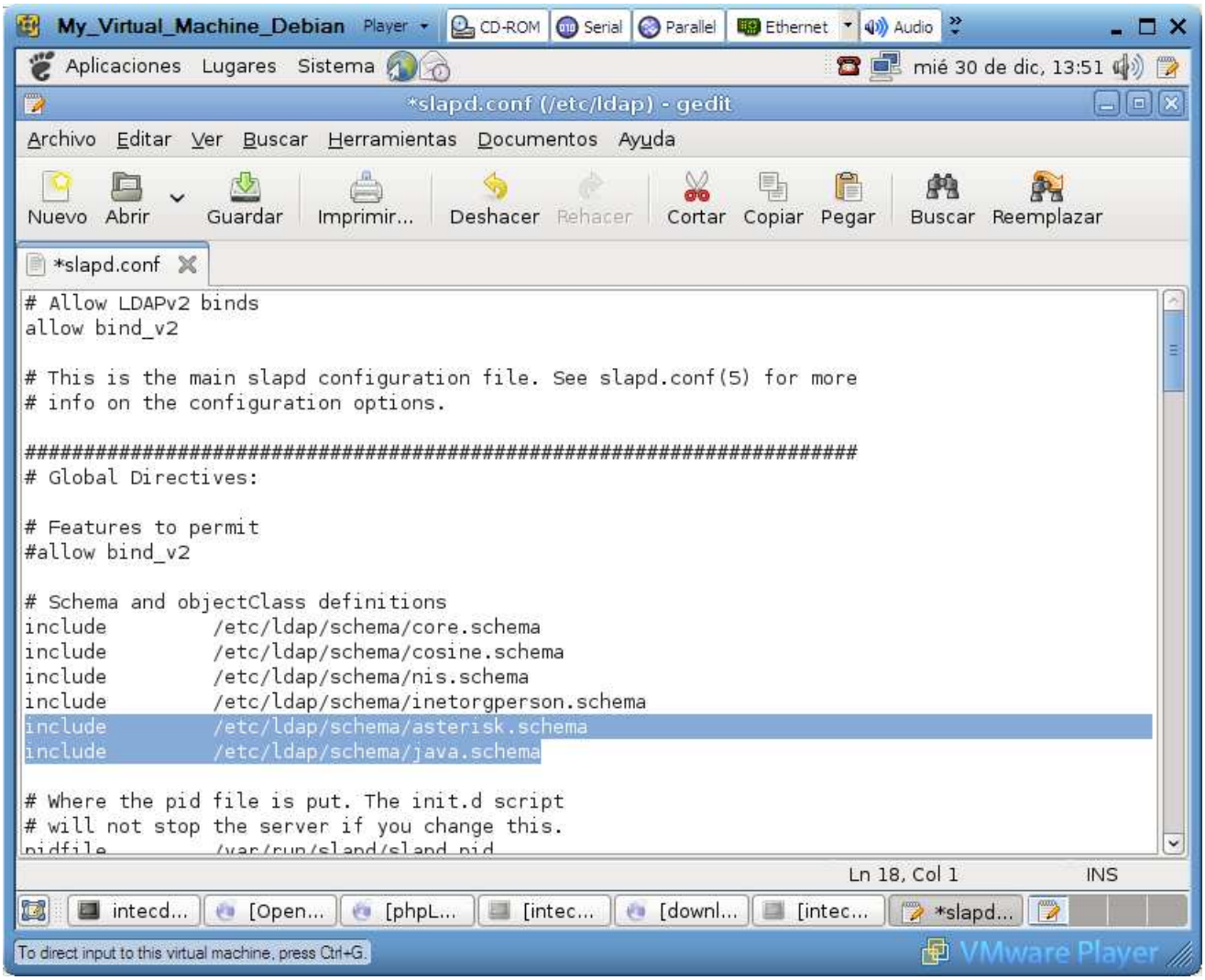


Incluir los esquemas de asterisk y java.
Descargar los esquemas de los siguientes enlaces y guardarlos en la carpeta /etc/ldap/schemas

Asterisk.schema:
http://www.voip-info.org/storage/users/992/22992/images/2069/asterisk.schema

java.schema:
www.bayour.com/openldap/schemas/java.schema

Incluir los esquemas en el fichero de configuración slapd.conf



4. Esquemas de LDAP

Tal como se comenta en el apartado LDAP de este documento, los directorios contienen unos elementos denominados esquemas. Estos esquemas definen el tipo de información que se almacena en el directorio, las reglas que debe cumplir dicha información y cómo se realizan las operaciones de búsqueda sobre los datos. Los esquemas pueden ser modificados para cubrir las necesidades específicas de un directorio.

Los esquemas básicos contienen un extenso conjunto de clases de objeto y atributos con los que se cumplen todas las necesidades de la mayoría de organizaciones. Se modelan siguiendo el estándar X.500 de la International Standards Organisation (ISO) para los servicios de directorio.

Los esquemas básicos utilizados en este proyecto son:

- Core.schema: esquema inicial básico de OpenLDAP. Es un esquema requerido.
- Nis.schema: Network Information Services schema. Para utilizar el directorio en red.
- InertOrgPerson.schema: frecuentemente utilizado para definir usuarios.

La siguiente tabla contiene los ficheros de esquema distribuidos por el proyecto OpenLDAP:

Table 8.1: Provided Schema Specifications

File	**Description**
`core.schema`	OpenLDAP *core* (required)
`cosine.schema`	Cosine and Internet X.500 (useful)
`inetorgperson.schema`	InetOrgPerson (useful)
`misc.schema`	Assorted (experimental)
`nis.schema`	Network Information Services (FYI)
`openldap.schema`	OpenLDAP Project (experimental)

Esquemas adicionales pueden ser incluidos en el directorio. En concreto, los siguientes esquemas han sido incluidos en la realización de este proyecto:

- Asterix.schema
- Java.schema

Ambos esquemas se adjuntan a continuación.

4.1. asterisk.schema

```
#
# Copyright (c) 2007-2008 Suretec Systems Ltd. - <http://www.suretecsystems.com/>
#
# Version: 3.1.5
#
# Changes:
# - Fixed wrong DESC - 07/05/2008
#
# Author: Gavin Henry - <ghenry@suretecsystems.com>
#
# Asterisk LDAP Schema
#
# Digium root OID (http://www.iana.org/assignments/enterprise-numbers)
#
#  1.3.6.1.4.1.22736
#  1.3.6.1.4.1.22736.5      LDAP elements
#  1.3.6.1.4.1.22736.5.4    Attribute Types
#  1.3.6.1.4.1.22736.5.5    Object Classes
#
objectIdentifier AsteriskRoot 1.3.6.1.4.1.22736
objectIdentifier AsteriskLDAP AsteriskRoot:5

################################################################################
# Attribute group OIDs.  e.g.: objectIdentifier AstAttrType AsteriskLDAP:4
################################################################################
objectIdentifier AstAttrType AsteriskLDAP:4


################################################################################
# Attribute OIDs e.g.: objectIdentifier AstContext AstAttrType:1
################################################################################
objectIdentifier AstContext AstAttrType:1
objectIdentifier AstExtension AstAttrType:2
objectIdentifier AstPriority AstAttrType:3
objectIdentifier AstApplication AstAttrType:4
objectIdentifier AstApplicationData AstAttrType:5
objectIdentifier AstAccountAMAFlags AstAttrType:6
objectIdentifier AstAccountCallerID AstAttrType:7
objectIdentifier AstAccountContext AstAttrType:8
objectIdentifier AstAccountMailbox AstAttrType:9
objectIdentifier AstMD5secret AstAttrType:10
objectIdentifier AstAccountDeny AstAttrType:11
objectIdentifier AstAccountPermit AstAttrType:12
objectIdentifier AstAccountQualify AstAttrType:13
objectIdentifier AstAccountType AstAttrType:14
objectIdentifier AstAccountDisallowedCodec AstAttrType:15
objectIdentifier AstAccountExpirationTimestamp AstAttrType:16
objectIdentifier AstAccountRegistrationContext AstAttrType:17
objectIdentifier AstAccountRegistrationExten AstAttrType:18
objectIdentifier AstAccountNoTransfer AstAttrType:19
objectIdentifier AstAccountCallGroup AstAttrType:20
objectIdentifier AstAccountCanReinvite AstAttrType:21
objectIdentifier AstAccountDTMFMode AstAttrType:22
objectIdentifier AstAccountFromUser AstAttrType:23
objectIdentifier AstAccountFromDomain AstAttrType:24
objectIdentifier AstAccountFullContact AstAttrType:25
objectIdentifier AstAccountHost AstAttrType:26
objectIdentifier AstAccountInsecure AstAttrType:27
objectIdentifier AstAccountNAT AstAttrType:28
objectIdentifier AstAccountPickupGroup AstAttrType:29
objectIdentifier AstAccountPort AstAttrType:30
```

```
objectIdentifier AstAccountRestrictCID AstAttrType:31
objectIdentifier AstAccountRTPTimeout AstAttrType:32
objectIdentifier AstAccountRTPHoldTimeout AstAttrType:33
objectIdentifier AstAccountRealmedPassword AstAttrType:34
objectIdentifier AstAccountAllowedCodec AstAttrType:35
objectIdentifier AstAccountMusicOnHold AstAttrType:36
objectIdentifier AstAccountCanCallForward AstAttrType:37
objectIdentifier AstAccountSecret AstAttrType:38
objectIdentifier AstAccountName AstAttrType:39
objectIdentifier AstConfigFilename AstAttrType:40
objectIdentifier AstConfigCategory AstAttrType:41
objectIdentifier AstConfigCategoryMetric AstAttrType:42
objectIdentifier AstConfigVariableName AstAttrType:43
objectIdentifier AstConfigVariableValue AstAttrType:44
objectIdentifier AstConfigCommented AstAttrType:45


################################################################################
# Object Class OIDs
################################################################################
objectIdentifier AstObjectClass AsteriskLDAP:2
objectIdentifier AsteriskExtension AstObjectClass:1
objectIdentifier AsteriskIAXUser AstObjectClass:2
objectIdentifier AsteriskSIPUser AstObjectClass:3
objectIdentifier AsteriskConfig AstObjectClass:4


################################################################################
# attribute definitions
#
# OID (the first arg) comes from the objectIdentifier defined above
#
# NAME should be the same as objectIdentifier
#
# DESC should be the description of the attribute
#
# EQUALITY is the rule to use when doing a search/compare for an
# attribute value.
#
# SUBSTR is the rule to use when doing a substring search (*foo*)
#
# SYNTAX is the syntax (i.e., type) of the attribute. We should
# probably stick to syntaxes:
#
#       1.3.6.1.4.1.1466.115.121.1.15   -> directoryString (UTF-8 string)
#       1.3.6.1.4.1.1466.115.121.1.26   -> IA5String (ASCII String)
#       1.3.6.1.4.1.1466.115.121.1.27   -> integer (Integer value)
#
# SINGLE-VALUE should be present if only one instance of this
# attribute is allowed within an entry.
#
# {32} is the allowed length
#
# e.g.:
#
# attributetype ( AstExample
#    NAME ( 'AstExample' )
#    DESC 'Asterisk Example Attribute'
#    EQUALITY caseIgnoreMatch
#    SUBSTR caseIgnoreSubstringsMatch
#    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32}
#    SINGLE-VALUE )
#
```

```
#########################################################################

attributetype ( AstContext
        NAME 'AstContext'
        DESC 'Asterisk Context'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstExtension
        NAME 'AstExtension'
        DESC 'Asterisk Extension'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstPriority
        NAME 'AstPriority'
        DESC 'Asterisk Priority'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstApplication
        NAME 'AstApplication'
        DESC 'Asterisk Application'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstApplicationData
        NAME 'AstApplicationData'
        DESC 'Asterisk Application Data'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountAMAFlags
        NAME 'AstAccountAMAFlags'
        DESC 'Asterisk Account AMA Flags'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountCallerID
        NAME 'AstAccountCallerID'
        DESC 'Asterisk Account CallerID'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountContext
        NAME 'AstAccountContext'
        DESC 'Asterisk Account Context'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountMailbox
        NAME 'AstAccountMailbox'
        DESC 'Asterisk Account Mailbox'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstMD5secret
        NAME 'AstMD5secret'
        DESC 'Asterisk Account MD5 Secret'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountDeny
        NAME 'AstAccountDeny'
        DESC 'Asterisk Account Deny'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountPermit
        NAME 'AstAccountPermit'
        DESC 'Asterisk Account Permit'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountQualify
        NAME 'AstAccountQualify'
        DESC 'Asterisk Account Qualify'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountType
        NAME 'AstAccountType'
        DESC 'Asterisk Account Type'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountDisallowedCodec
        NAME 'AstAccountDisallowedCodec'
        DESC 'Asterisk Account Disallowed Codec'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountExpirationTimestamp
        NAME 'AstAccountExpirationTimestamp'
        DESC 'Asterisk Account Expiration Timestamp'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRegistrationContext
        NAME 'AstAccountRegistrationContext'
        DESC 'Asterisk Account Registration Context'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRegistrationExten
        NAME 'AstAccountRegistrationExten'
        DESC 'Asterisk Account Registration Extension'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountNoTransfer
        NAME 'AstAccountNoTransfer'
        DESC 'Asterisk Account No Transfer'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountCallGroup
        NAME 'AstAccountCallGroup'
        DESC 'Asterisk Account Call Group'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountCanReinvite
        NAME 'AstAccountCanReinvite'
        DESC 'Asterisk Account Can Reinvite'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountDTMFMode
        NAME 'AstAccountDTMFMode'
        DESC 'Asterisk Account DTMF Flags'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountFromUser
        NAME 'AstAccountFromUser'
        DESC 'Asterisk Account From User'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountFromDomain
        NAME 'AstAccountFromDomain'
        DESC 'Asterisk Account From Domain'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountFullContact
        NAME 'AstAccountFullContact'
        DESC 'Asterisk Account Full Contact'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountHost
        NAME 'AstAccountHost'
        DESC 'Asterisk Account Host'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountInsecure
        NAME 'AstAccountInsecure'
        DESC 'Asterisk Account Insecure'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

```
attributetype ( AstAccountNAT
        NAME 'AstAccountNAT'
        DESC 'Asterisk Account NAT'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountPickupGroup
        NAME 'AstAccountPickupGroup'
        DESC 'Asterisk Account PickupGroup'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountPort
        NAME 'AstAccountPort'
        DESC 'Asterisk Account Port'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRestrictCID
        NAME 'AstAccountRestrictCID'
        DESC 'Asterisk Account Restrict CallerID'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRTPTimeout
        NAME 'AstAccountRTPTimeout'
        DESC 'Asterisk Account RTP Timeout'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRTPHoldTimeout
        NAME 'AstAccountRTPHoldTimeout'
        DESC 'Asterisk Account RTP Hold Timeout'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountRealmedPassword
        NAME 'AstAccountRealmedPassword'
        DESC 'Asterisk Account Realmed Password'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountAllowedCodec
        NAME 'AstAccountAllowedCodec'
        DESC 'Asterisk Account Allowed Codec'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountMusicOnHold
        NAME 'AstAccountMusicOnHold'
        DESC 'Asterisk Account Music On Hold'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

```
attributetype ( AstAccountCanCallForward
        NAME 'AstAccountCanCallForward'
        DESC 'Asterisk Account Can Call Forward'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountSecret
        NAME 'AstAccountSecret'
        DESC 'Asterisk Account Secret'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstAccountName
        NAME 'AstAccountName'
        DESC 'Asterisk Account Username'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigFilename
        NAME 'AstConfigFilename'
        DESC 'Asterisk LDAP Configuration Filename'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigCategory
        NAME 'AstConfigCategory'
        DESC 'Asterisk LDAP Configuration Category'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigCategoryMetric
        NAME 'AstConfigCategoryMetric'
        DESC 'Asterisk LDAP Configuration Category Metric'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigVariableName
        NAME 'AstConfigVariableName'
        DESC 'Asterisk LDAP Configuration Variable Name'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigVariableValue
        NAME 'AstConfigVariableValue'
        DESC 'Asterisk LDAP Configuration Variable Value'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype ( AstConfigCommented
        NAME 'AstConfigCommented'
        DESC 'Asterisk LDAP Configuration Commented'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

```
##############################################################################
# Object Class definitions
#
# This is where to define the object classes. Object classes are used
# to define which attribute MAY (optional) or MUST (required) belong
# to an entry.
#
# Classes can be AUXILIARY or STRUCTURAL. An entry in the directory
# must have one and only one structural class, but can have many
# AUXILIARY classes.
#
##############################################################################

objectclass ( AsteriskExtension
    NAME 'AsteriskExtension'
    DESC 'PBX Extension Information for Asterisk'
    SUP top AUXILIARY
    MUST cn
    MAY (
        AstContext $
        AstExtension $
        AstPriority $
        AstApplication $
        AstApplicationData
    )
    )

##############################################################################
#
# AsteriskIAXUser and AsteriskSIPUser extend AsteriskExtension. These
# represent real accounts in Asterisk.
#
# NOTE: They are defined as AUXILIARY in case they need to be mixed with an
# existing directory deployment.
#
##############################################################################

objectclass ( AsteriskIAXUser
    NAME 'AsteriskIAXUser'
    DESC 'IAX2 User information for Asterisk'
    SUP AsteriskExtension AUXILIARY
    MUST cn
    MAY (
        AstAccountAMAFlags $
        AstAccountCallerID $
        AstAccountContext $
        AstAccountFullContact $
        AstAccountHost $
        AstAccountMailbox $
        AstMD5secret $
        AstAccountDeny $
        AstAccountPermit $
        AstAccountPort $
        AstAccountQualify $
        AstAccountType $
        AstAccountDisallowedCodec $
        AstAccountExpirationTimestamp $
        AstAccountRegistrationContext$
        AstAccountRegistrationExten $
        AstAccountNoTransfer $
        AstAccountName
    )
    )
```

```
objectclass ( AsteriskSIPUser
    NAME 'AsteriskSIPUser'
    DESC 'SIP User information for Asterisk'
    SUP AsteriskExtension AUXILIARY
    MUST cn
    MAY (
        AstAccountAMAFlags $
        AstAccountCallGroup $
        AstAccountCallerID $
        AstAccountCanReinvite $
        AstAccountContext $
        AstAccountDTMFMode $
        AstAccountFromUser $
        AstAccountFromDomain $
        AstAccountFullContact $
        AstAccountHost $
        AstAccountInsecure $
        AstAccountMailbox $
        AstAccountRealmedPassword $
        AstAccountNAT $
        AstAccountDeny $
        AstAccountPermit $
        AstAccountPickupGroup $
        AstAccountPort $
        AstAccountQualify $
        AstAccountRestrictCID $
        AstAccountRTPTimeout $
        AstAccountRTPHoldTimeout $
        AstAccountType $
        AstAccountDisallowedCodec $
        AstAccountAllowedCodec $
        AstAccountMusicOnHold $
        AstAccountExpirationTimestamp $
        AstAccountRegistrationContext $
        AstAccountRegistrationExten $
        AstAccountCanCallForward $
        AstAccountSecret $
        AstAccountName
    )
    )
###############################################################################
#
# AsteriskIAXUser and AsteriskSIPUser extend AsteriskExtension. These
# represent real accounts in Asterisk.
#
# NOTE: They are defined as AUXILIARY in case they need to be mixed with an
# existing directory deployment.
#
###############################################################################
objectclass ( AsteriskConfig
    NAME 'AsteriskConfig'
    DESC 'Asterisk configuration Information'
    SUP top AUXILIARY
    MUST cn
    MAY (
        AstConfigFilename $
        AstConfigCategory $
        AstConfigCategoryMetric $
        AstConfigVariableName $
        AstConfigVariableValue $
        AstConfigCommented
    )
    )
```

4.2. java.schema

```
-- Attribute types --

( 1.3.6.1.4.1.42.2.27.4.1.6
  NAME 'javaClassName'
  DESC 'Fully qualified name of distinguished Java class or interface'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
)

( 1.3.6.1.4.1.42.2.27.4.1.7
  NAME 'javaCodebase'
  DESC 'URL(s) specifying the location of class definition'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
)

( 1.3.6.1.4.1.42.2.27.4.1.8
  NAME 'javaSerializedData'
  DESC 'Serialized form of a Java object'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
  SINGLE-VALUE
)

( 1.3.6.1.4.1.42.2.27.4.1.10
  NAME 'javaFactory'
  DESC 'Fully qualified Java class name of a JNDI object factory'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
)

( 1.3.6.1.4.1.42.2.27.4.1.11
  NAME 'javaReferenceAddress'
  DESC 'Addresses associated with a JNDI Reference'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

( 1.3.6.1.4.1.42.2.27.4.1.12
  NAME 'javaDoc'
  DESC 'The Java documentation for the class'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
)

( 1.3.6.1.4.1.42.2.27.4.1.13
  NAME 'javaClassNames'
  DESC 'Fully qualified Java class or interface name'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

-- from RFC-2256 --

( 2.5.4.13
  NAME 'description'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
```

```
)

-- Object classes --

( 1.3.6.1.4.1.42.2.27.4.2.1
  NAME 'javaContainer'
  DESC 'Container for a Java object'
  SUP top
  STRUCTURAL
  MUST ( cn )
)

( 1.3.6.1.4.1.42.2.27.4.2.4
  NAME 'javaObject'
  DESC 'Java object representation'
  SUP top
  ABSTRACT
  MUST ( javaClassName )
  MAY ( javaClassNames $ javaCodebase $ javaDoc $ description )
)

( 1.3.6.1.4.1.42.2.27.4.2.5
  NAME 'javaSerializedObject'
  DESC 'Java serialized object'
  SUP javaObject
  AUXILIARY
  MUST ( javaSerializedData )
)

( 1.3.6.1.4.1.42.2.27.4.2.7
  NAME 'javaNamingReference'
  DESC 'JNDI reference'
  SUP javaObject
  AUXILIARY
  MAY ( javaReferenceAddress $ javaFactory )
)

( 1.3.6.1.4.1.42.2.27.4.2.8
  NAME 'javaMarshalledObject'
  DESC 'Java marshalled object'
  SUP javaObject
  AUXILIARY
  MUST ( javaSerializedData )
)

-- Matching rule from ISO X.520 --

( 2.5.13.5
  NAME 'caseExactMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)
```

5. Servidor conteniendo la instalación de la aplicación de gestión de usuarios

Para el desarrollo de este proyecto no ha sido necesaria la creación de un servidor conteniendo la aplicación de gestión de usuarios, sino que tal como se ha detallado, se ha accedido al servidor de desarrollo de aplicaciones de la empresa Intecdom para acceder al repositorio central, integrar el nuevo sofware desarrollado en una “rama” de desarrollo del proyecto.

En el caso de una futura implantación de este producto, dicha instalación vendría preinstalada en un elemento hardware proporcionado por la empresa distribuidora de la centralita telefónica IRI, por lo que no se detallarán los pasos unitarios de instalación de cada parte constitutiva de un servidor IRI.

Se detallará a continuación los diversos elementos de los que consta la aplicación de gestión de usuarios:

- Sistema Operativo Debian
- Asterisk
- Apache
- Java
- PostgreSQL
- Jboss
- Paquetes de Software de la IRI